Team: L2G2

Eric Yang – Project Manager

Joseph Buskirk - Developer

Gurdev Singh – Developer

Jonathan Wong- UI/UX

Romeo Vanegas - UI/UX

Version: 1.0

Fresno state class info

Software Design Document

**Revision History**

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Description** |
| December 2, 2022 | 1.0 | Initial Version of Document |

**Table of Contents**

**1.** [**Introduction**](#Introduction) **…………………………………………………………………………………………….……… 3**

**1.1 Purpose ………………………………………………………………………………………….……… 3**

**1.2 Scope ……………………………………………………………………………………………..……… 3**

**1.3 Overview .……………………………………………………………………………………….……. 3**

**2.** [**Use Cases**](#UseCase)  **.………………………………………………………………………………………………..……… 4**

**2.1 Actors .……………………………………………………………………………………….……****……. 4**

**2.2 List of Use Cases .…………………………………………………………………………….……. 4**

**2.3 Use Case Diagrams .……………………………………………………………………………..… 5**

**3.** [**Design Overview**](#DesignOverview) **.………………………………………………………………………………………….. 6**

**3.1 Introduction .…………………………………………………………………………………………. 6**

**3.2 System Architecture .…………………………………………………………………………….. 6**

**3.3 System Interfaces .…………………………………………………………………………………. 6**

**3.4 Constraints and Assumptions .………………………………………………………… ……… 7**

**4.** [**Object Description**](#ObjectDesc) **.………………………………………………………………………………………… 8**

**4.1 Objects .………………………………………………………………………………………………….. 8**

**5.** [**Dynamic Model**](#Dynamic) **.……………………………………………………………………………………………… 31**

**5.1 Sequence Diagrams ………………………………………………………………………………… 31**

**5.2 State Diagrams …………………………………………………………………………………….… 34**

**6.** [**Non-Functional Requirements**](#Reqs) **……………………………………………………………………. 36**

**6.1 Performance Requirements …………………………………………………………………… 36**

**6.2 Design Constraints …………………………………………………………………………………. 36**

**1. Introduction**

Fresno State Class Info (FSCI) serves as an extended catalog for California State University, Fresno’s classes. Many Fresno State students find it difficult to obtain information about their classes or professors because the current catalog has lacked a lot of the information that is needed. This lack of information has lead students not being prepared for their classes.

The goal of this service is to increase the amount of data that is being given to each student and achieve a much more efficient way foreach student to obtain the information.

**1.1 Purpose**

The purpose of this document is to define the architecture and system design of this project in order to understand what is expected of this software to accomplish. This Software Design Document will provide the information necessary to provide descriptions and details for the software and system to be built.

**1.2 Scope**

This Software Design Document will provide enough descriptions for the software and its systems to be built on a basic level of functionality. This document will be focused on a base leveled system and its main features.

**1.3 Overview**

This Software Design Document will be divided into 8 different sections.

1. Introduction
2. Use Cases
3. Design Overview
4. Object Description
5. Object Collaboration
6. Data Design
7. Dynamic Model
8. Non-Functional Requirements

**2.** **Use Cases**

**2.1 Actors**

**2.1.1 Public Users**

All of these actors will be able to use all the features that are given to them. The difference between each of the actors would be the reasons to use this software. They will all be able to search for classes and professors and access information according to what they wanted to look for.

**2.1.1.1 Students**

**2.1.1.2 Professors**

The students will represent all Fresno State students that want to use this website. These users are going to be the vast majority of the users that are on this website.

Professors may want to use with website if they want to access an old syllabus or link that they have provided in the past. Students will also be able to access these valuable links.

**2.1.2** **Administrative Users**

These users will be able to edit the database that contains all of the information for each class and professor. They will be overseeing and working with the data being provided.

**2.2 List of Use Cases**

**2.2.1 Public User Use Cases**

**2.2.1.1** Search for Classes / Professors using a “Search Bar”

**2.2.1.2** Search for Classes using a “Department” Tab

**2.2.1.3** Search for Professorsusing a “Faculties” Tab

**2.2.2 Administrative User Use Cases**

**2.2.2.1** Add data to a Class / professor

**2.2.2.2** Edit data in a Class / professor

**2.2.2.3** Delete data in a Class / professor

**2.3 Use Case Diagrams**

**2.3.1 Public Users**

**Diagram

Description automatically generated**

**2.3.2 Administrative Users**

**Diagram

Description automatically generated**

**3.** **Design Overview**

**3.1 Introduction**

The system’s architecture mainly consists of a dynamic. Behind this interface, there will be functions to change what is being displayed and functions to grab data from the database.

**3.2 System Architecture**

**Diagram

Description automatically generated**

**3.3 System Interfaces**

**3.3.1 User Interface**

The user interface will allow users to easily navigate through the website. Users will be able to use all navigational functions based on how they want to navigate through the website. This will communicate to the database using the software interface.

**3.3.2 Software Interface**

This interface will need to pull data from the database and push it to the user interface where it will then be displayed. This will act as the communicator between the database holding all the information and the user using the user interface.

**3.4 Constraints and Assumptions**

**3.4.1 Assumptions**

**3.4.1.1** Users are able to use the internet to access the website and its database server.

**3.4.1.2** The system is bug free.

**3.4.1.3** All of the supporting information for each class and professor is accurate.

**3.4.2 Constraints**

**3.4.2.1** This iteration of the website is now dependent on the database.

**3.4.2.2** There must be at least one entry inside of the database for the communication between interfaces remains unaffected and running correctly.

**4.** **Object Description**

**4.1 Objects**

**4.1.1 User Interface (index.html)**

|  |  |
| --- | --- |
| **Class Name:** index.html | |
| **Description:** This is the dynamic home page that will load everything for us. It will include the corresponding classes and allow everything to be displayed.  This html page will be used as the user interface. No other pages will be used as a user interface.  This page does not have function but will have basic inputs so that it can control what is being done on the page. The functions are included from other files. | |
| **Tags** | **Tag Description** |
| div 🡪 class = “topbar” | This will contain all of the navigational options for this dynamic page. It contains the Logo, “About Us”, “Departments”, and “Faculty” tabs. These all will allow users to navigate throughout the page. There is also a search bar the allows users to type in class or professor names. |
| div 🡪 class = “container” | This container will essentially house everything that will load onto the page. This is the only portion of the page that will change as well. Everything will be loaded here dynamically. |
| div 🡪 class = “bottombar” | This will contain basic information about the page and about this team. It will also contain copyright information and the date. |

**4.1.2 Software (scripts.js)**

|  |  |
| --- | --- |
| **Class Name:** scripts.js | |
| **Description:** This is the file that is read throughout “index.html”. This class contains all of the functionalities that the home page will use.  This will pull data from the database and parse it accordingly. It will then manipulate that data based on what was parsed and correctly display it.  This class will directly change the html of the home page, adding, deleting, and editing the page. This will allow the page to be completely dynamic and avoid creating hundreds of pages for each class and professor. | |
| **Function** | **Function Description** |
| autocomplete(inp, arr) | This function takes in an input and an array. By doing this, and applying it to the search bar, the search bar is able to complete a search for the user. This will help ease trying to find classes and shorten the amount of work the user needs to do. |
| **Program Description** |
| function autocomplete(inp, arr) {      /\*the autocomplete function takes two arguments,      the text field element and an array of possible autocompleted values:\*/      var currentFocus;      /\*execute a function when someone writes in the text field:\*/      inp.addEventListener("input", function(e) {          var a, b, i, val = this.value;          /\*close any already open lists of autocompleted values\*/          closeAllLists();          if (!val) { return false;}          currentFocus = -1;          /\*create a DIV element that will contain the items (values):\*/          a = document.createElement("DIV");          a.setAttribute("id", this.id + "autocomplete-list");          a.setAttribute("class", "autocomplete-items");          /\*append the DIV element as a child of the autocomplete container:\*/          this.parentNode.appendChild(a);          /\*for each item in the array...\*/          for (i = 0; i < arr.length; i++) {              /\*check if the item starts with the same letters as the text field value:\*/              if (arr[i].substr(0, val.length).toUpperCase() == val.toUpperCase()) {                  /\*create a DIV element for each matching element:\*/                  b = document.createElement("DIV");                  /\*make the matching letters bold:\*/                  b.innerHTML = "<strong>" + arr[i].substr(0, val.length) + "</strong>";                  b.innerHTML += arr[i].substr(val.length);                  /\*insert a input field that will hold the current array item's value:\*/                  b.innerHTML += "<input type='hidden' value='" + arr[i] + "'>";                  /\*execute a function when someone clicks on the item value (DIV element):\*/                      b.addEventListener("click", function(e) {                    /\*insert the value for the autocomplete text field:\*/                      inp.value = this.getElementsByTagName("input")[0].value;                      /\*close the list of autocompleted values,                      (or any other open lists of autocompleted values:\*/                      closeAllLists();                  });                  a.appendChild(b);              }          }      });      /\*execute a function presses a key on the keyboard:\*/      inp.addEventListener("keydown", function(e) {          var x = document.getElementById(this.id + "autocomplete-list");          if (x) x = x.getElementsByTagName("div");          if (e.keyCode == 40) {              /\*If the arrow DOWN key is pressed,              increase the currentFocus variable:\*/              currentFocus++;              /\*and and make the current item more visible:\*/              addActive(x);              } else if (e.keyCode == 38) { //up              /\*If the arrow UP key is pressed,              decrease the currentFocus variable:\*/              currentFocus--;              /\*and and make the current item more visible:\*/              addActive(x);              } else if (e.keyCode == 13) {              /\*If the ENTER key is pressed, prevent the form from being submitted,\*/              e.preventDefault();              if (currentFocus > -1) {                  /\*and simulate a click on the "active" item:\*/                  if (x) x[currentFocus].click();              }          }      });      function addActive(x) {          /\*a function to classify an item as "active":\*/          if (!x) return false;          /\*start by removing the "active" class on all items:\*/          removeActive(x);          if (currentFocus >= x.length) currentFocus = 0;          if (currentFocus < 0) currentFocus = (x.length - 1);          /\*add class "autocomplete-active":\*/          x[currentFocus].classList.add("autocomplete-active");      }      function removeActive(x) {          /\*a function to remove the "active" class from all autocomplete items:\*/          for (var i = 0; i < x.length; i++) {              x[i].classList.remove("autocomplete-active");          }      }      function closeAllLists(elmnt) {          /\*close all autocomplete lists in the document,          except the one passed as an argument:\*/          var x = document.getElementsByClassName("autocomplete-items");          for (var i = 0; i < x.length; i++) {              if (elmnt != x[i] && elmnt != inp) {                  x[i].parentNode.removeChild(x[i]);              }          }      }      /\*execute a function when someone clicks somewhere else in the document:\*/      document.addEventListener("click", function (e) {          closeAllLists(e.target);      });  } |
| findSearch() | This function will read the given input from the search bar and compare it with the database. If it is found then it will display the class or professor page. If the name of the class or professor was not found then it will alert the user that the search was not found.  It requires the full name of the class or professor to properly return the page that the user is looking for. Partial names will not work for this search engine. |
| **Program Description** |
| function findSearch() {      let searchInput = document.getElementById("searchBar").value;      let i = 0      let max = 0;      if (data[0]["rows"].length > nameProfArr.length) {          max = data[0]["rows"].length;      }      else {          max = nameProfArr.length;      }      for (i; i < max; i++) {          if (searchInput == data[0]["rows"][i][0]) {              classIndex = i;              displayClassInfoClass();              loadClassTemplate();              break;          }          if (searchInput == nameProfArr[i]) {              profIndex = i;              displayProfInfo();              loadProfTemplate();              break;          }      }      if (i == max) {          alert("Search Not Found!");      }  } |
| deleteAll() | This function grabs all the data from the “container” from index.html and empties it. This will be used in almost all display functions. This helps simplify clearing the screen of data. It will not affect the top navigational bar and the footer at the bottom. |
| **Program Description** |
| function deleteAll(){      let bod = document.getElementById("container");      bod.innerHTML = '';  } |
| aboutUs() | This function create the “About Us” page. This function is called when the tab “About Us” is clicked. It will display data about this team. |
| **Program Description** |
| function aboutUs(){      deleteAll();      let bod = document.getElementById("container");          let tempContainer = document.createElement('div');              tempContainer.id = "fakeGrid";          let newDiv = document.createElement('div');              newDiv.className = "aboutUsPicOne";              newDiv.style.width = "100px"              let ericImg1 = document.createElement('img');                  ericImg1.src = aboutUsData["Eric"]["Pic1"];                  ericImg1.id = "aboutUsPic1"                  ericImg1.onclick = function () {                      aboutUsIndex = "Eric";                      displayAboutPerson();                  };              let gurdevImg1 = document.createElement('img');                  gurdevImg1.src = aboutUsData["Gurdev"]["Pic1"];                  gurdevImg1.id = "aboutUsPic1"                  gurdevImg1.onclick = function () {                      aboutUsIndex = "Gurdev";                      displayAboutPerson();                  };              let JosephImg1 = document.createElement('img');                  JosephImg1.src = aboutUsData["Joseph"]["Pic1"];                  JosephImg1.id = "aboutUsPic1"                  JosephImg1.onclick = function () {                      aboutUsIndex = "Joseph";                      displayAboutPerson();                  };              let JohnathanImg1 = document.createElement('img');                  JohnathanImg1.src = aboutUsData["Johnathan"]["Pic1"];                  JohnathanImg1.id = "aboutUsPic1"                  JohnathanImg1.onclick = function () {                      aboutUsIndex = "Johnathan";                      displayAboutPerson();                  };              let romeoImg1 = document.createElement('img');                  romeoImg1.src = aboutUsData["Romeo"]["Pic1"];                  romeoImg1.id = "aboutUsPic1"                  romeoImg1.onclick = function () {                      aboutUsIndex = "Romeo";                      displayAboutPerson();                  };          newDiv.appendChild(ericImg1);          newDiv.appendChild(gurdevImg1);          newDiv.appendChild(JosephImg1);          newDiv.appendChild(JohnathanImg1);          newDiv.appendChild(romeoImg1);          tempContainer.appendChild(newDiv)      bod.appendChild(tempContainer);      displayAboutPerson();  } |
| deleteDisplayCurrentClass() | This will delete the current class’s information. This is used when tabbing through the data being presented by the classes. |
| **Program Description** |
| function deleteDisplayCurrentClass(){      let bod = document.getElementById("lowerBodyClasses");      if(typeof(bod) != 'undefined' && bod != null){          bod.innerHTML = '';      }  } |
| deleteDisplayCurrentProf() | This will delete the current professor’s information. This is used when tabbing through the data being presented for the professor. |
| **Program Description** |
| function deleteDisplayCurrentProf(){      let bod = document.getElementById("lowerBodProfs");      if(typeof(bod) != 'undefined' && bod != null){          bod.innerHTML = '';      }  } |
| displayProfInfo() | This will display all of the basic information for the professor that is currently being displayed. |
| **Program Description** |
| function displayProfInfo() {      deleteDisplayCurrentClass();      deleteDisplayCurrentProf();      let newProf = document.createElement('div');          newProf.id = "professor";          newProf.className = "tabcontent";      let newH3F = document.createElement('h3');          newH3F.appendChild(document.createTextNode(getProfDepartment(profIndex)));      newProf.appendChild(newH3F);      let newH3A = document.createElement('p');          newH3A.appendChild(document.createTextNode("Staff/Faculty Role: " + getProfRole(profIndex)));      newProf.appendChild(newH3A);      let newH3B = document.createElement('p');          newH3B.appendChild(document.createTextNode("Desk Phone Number: " + getProfDeskPhone(profIndex)));      newProf.appendChild(newH3B);      let newH3C = document.createElement('p');          newH3C.appendChild(document.createTextNode("Department Phone Number: " + getProfDepartMentPhone(profIndex)));      newProf.appendChild(newH3C);      let newH3D = document.createElement('p');          newH3D.appendChild(document.createTextNode("Email: " + getProfEmail(profIndex)));      newProf.appendChild(newH3D);      let newH3E = document.createElement('p');          newH3E.appendChild(document.createTextNode("Office Location: " + getProfOfficeLoc(profIndex)));      newProf.appendChild(newH3E);      let newH3G = document.createElement('p');          newH3G.appendChild(document.createTextNode("Mail Stop Location: " + getProfMailStop(profIndex)));      newProf.appendChild(newH3G);      lowerBodProfs.appendChild(newProf);  } |
| displaySyllabusClass() | This displays the syllabus for the class that is currently being displayed. |
| **Program Description** |
| function displaySyllabusClass() {      deleteDisplayCurrentClass();      deleteDisplayCurrentProf();      let newEmbSyll = document.createElement('embed');          newEmbSyll.src = getCsciSyllabus(classIndex);      if (newEmbSyll.src.substring(newEmbSyll.src.length-9, newEmbSyll.src.length) != "undefined") {          let newSyll = document.createElement('div');              newSyll.id = "syllabus";              newSyll.className = "tabcontent";                  newEmbSyll.style.width = "100%";                  newEmbSyll.style.height = "800px";                  newEmbSyll.style.alignContent = "center";              newSyll.appendChild(newEmbSyll);          lowerBodClasses.appendChild(newSyll);      }      else {          let newSyll = document.createElement('div');              newSyll.id = "syllabus";              newSyll.className = "tabcontent";              let unavailable = document.createElement('p');              unavailable.appendChild(document.createTextNode("Syllabus unavailable at this time"));              newSyll.appendChild(unavailable);              lowerBodClasses.appendChild(newSyll);      }  } |
| displayClassInfoClass() | This displays the basic information for the class being displayed. |
| **Program Description** |
| function displayClassInfoClass(){      deleteDisplayCurrentClass();      deleteDisplayCurrentProf();      let classUnits = document.createElement('p');          classUnits.id = "classDescript";          classUnits.appendChild(document.createTextNode(getUnits(classIndex)));      let classOffered = document.createElement('p');          classOffered.id = "classDescript";          if(getOfferedIn(classIndex) == ""){              classOffered.appendChild(document.createTextNode("Only Offered Occasionally"));          }          else{              classOffered.appendChild(document.createTextNode("Typically Offered in: " + getOfferedIn(classIndex)));          }      let classAddInfo = document.createElement('p');          classAddInfo.id = "classDescript";          classAddInfo.appendChild(document.createTextNode("Additional Information: " + getAdditionalInfo(classIndex)));      let classDescript = document.createElement('p');          classDescript.id = "classDescript";          classDescript.appendChild(document.createTextNode("Description: " + getDescription(classIndex)));      lowerBodClasses.appendChild(classUnits);      lowerBodClasses.appendChild(classOffered);      lowerBodClasses.appendChild(classAddInfo);      if(getAdditionalInfo(classIndex) == ""){          classAddInfo.remove();      }      lowerBodClasses.appendChild(classDescript);  } |
| displayClassesProfs() | This displays all classes that the current professor being presented has taught or is teaching. |
| **Program Description** |
| function displayClassesProfs(){      deleteDisplayCurrentClass();      deleteDisplayCurrentProf();      let newDesc = document.createElement('div');          newDesc.id = "description";          newDesc.className = "tabcontent";          let list = getCsciList(profIndex);          for(let i = 0; i < list.length; i++){              let p = document.createElement('p');              p.className = "PartOfClassList";              p.onclick = function () {                  for (let j = 0; j < data[0]["rows"].length; j++) {                      if (list[i] == data[0]["rows"][j][0]) {                          classIndex = j;                          displayClassInfoClass();                          loadClassTemplate();                          break;                      }                  }              };              p.appendChild(document.createTextNode(list[i]));              newDesc.appendChild(p);          }      lowerBodProfs.appendChild(newDesc);  } |
| displayAdditionalLinksProf() | This displays the current professor’s additional links. These links will be anchors that move users to different websites. |
| **Program Description** |
| function displayAdditionalLinksProf(){      deleteDisplayCurrentClass();      deleteDisplayCurrentProf();      let newLink = document.createElement('div');          newLink.id = "addLinks";          newLink.className = "tabcontent";          let newList = document.createElement('ul')          let list = getCsciLinks(profIndex)          for(let i = 0; i < list.length; i++){              let li = document.createElement('li');              let a = document.createElement('a');              a.appendChild(document.createTextNode(list[i]));              a.href = list[i];              a.target = "\_blank";              li.appendChild(a);              newList.appendChild(li);          }          newLink.appendChild(newList);      lowerBodProfs.appendChild(newLink);  } |
| loadClassTemplate() | This is how the classes will be loaded. They will have 3 tabs that will help the user navigate through the page and get to where they would need to, to gather information. It will have a basic information page, a syllabus page, and additional links. |
| **Program Description** |
| function loadClassTemplate(){      deleteAll();      let bod = document.getElementById("container");      //Start of flex-container      let flexContainer = document.createElement('div');          flexContainer.className = "flex-container";          //Start of tab          let tab = document.createElement('div');              tab.className = "tab";              let tablink1 = document.createElement('button');                  tablink1.className = "tablinks";                  tablink1.onclick = displayClassInfoClass;                  tablink1.appendChild(document.createTextNode("Class Description"));              let tablink2 = document.createElement('button');                  tablink2.className = "tablinks";                  tablink2.onclick = displaySyllabusClass;                  tablink2.appendChild(document.createTextNode("Syllabus"));              let tablink3 = document.createElement('button');                  tablink3.className = "tablinks";                  tablink3.onclick = displayAdditionalLinksClass;                  tablink3.appendChild(document.createTextNode("Additional Links"));              tab.appendChild(tablink1);              tab.appendChild(tablink2);              tab.appendChild(tablink3);          //End of tab           //Start of classInfo          let classInfo = document.createElement('div');              classInfo.id = "classInfo";              let flexContainerRow = document.createElement('div'); //Start of flexContainerRow                  flexContainerRow.className = "flex-containerRow";                  let classTitle = document.createElement('div');                      let temp1H1 = document.createElement('h1');                          temp1H1.appendChild(document.createTextNode(createShortName(data[0]["rows"][classIndex][0])))                      classTitle.className = "classTitle";                      classTitle.appendChild(temp1H1);                  let classTitle2 = document.createElement('div');                      let temp2H1 = document.createElement('h1');                          temp2H1.appendChild(document.createTextNode(createClassName(data[0]["rows"][classIndex][0])))                      classTitle2.className = "classTitle2";                      classTitle2.appendChild(temp2H1);                  flexContainerRow.appendChild(classTitle);                  flexContainerRow.appendChild(classTitle2);              classInfo.appendChild(flexContainerRow);          //End of classInfo          flexContainer.appendChild(tab);          flexContainer.appendChild(classInfo);          flexContainer.appendChild(lowerBodClasses);      //End of flex-container      bod.appendChild(flexContainer);  } |
| loadProfTemplate() | This will load the professor pages onto the screen dynamically. 3 tabs will be created for further navigation per professor page: basic information, class list, and additional links. |
| **Program Description** |
| function loadProfTemplate(){      deleteAll();      displayProfInfo();      let container = document.getElementById("container");      let bod = document.createElement("div");          bod.className = "flex-container";          bod.id = "flex-container";          let newTabBar = document.createElement('div');              newTabBar.className = "tab";              newTabBar.style.backgroundColor = "#13284c";              let profButton = document.createElement('button');                  profButton.className = "profLinks";                  profButton.onclick = displayProfInfo;                  profButton.value = "Professor Information";                  profButton.appendChild(document.createTextNode("Professor Information"));              let descButton = document.createElement('button');                  descButton.className = "profLinks";                  descButton.onclick = displayClassesProfs;                  descButton.value = "Class List";                  descButton.appendChild(document.createTextNode("Class List"));              // let syllButton = document.createElement('button');              //  syllButton.className = "profLinks";              //  syllButton.onclick = displaySyllabusProf;              //  syllButton.value = "Syllabus";              //  syllButton.appendChild(document.createTextNode("Syllabus"));              let linkButton = document.createElement('button');                  linkButton.className = "profLinks";                  linkButton.onclick = displayAdditionalLinksProf;                  linkButton.value = "Additional Links";                  linkButton.appendChild(document.createTextNode("Additional Links"));              newTabBar.appendChild(profButton);              newTabBar.appendChild(descButton);              //newTabBar.appendChild(syllButton);              newTabBar.appendChild(linkButton);          bod.appendChild(newTabBar);          let newContent = document.createElement('div');              newContent.className = "tabContent";              newContent.id = "professor";              let newParent = document.createElement('div');                  newParent.className = "parent";                  let newImg = document.createElement('img');                      newImg.src = csciProfImages[profIndex];                      newImg.className = "upload-icon";                  newParent.appendChild(newImg);              newContent.appendChild(newParent);          bod.appendChild(newContent);          let newH1 = document.createElement('h1');              newH1.appendChild(document.createTextNode(nameProfArr[profIndex]));              newH1.style.alignContent = "center";          bod.appendChild(newH1);          bod.appendChild(lowerBodProfs);      container.appendChild(bod);  } |
| displayFacultyPage() | This will display the faculty page when “Faculty” is clicked on the home page. This function currently only displays the Computer Science faculty.  It will display the image of the faculty member, name, role, and their contact information.  Each cell of the table that was created by this function will also allow you to navigate to the appropriate professor page. |
| **Program Description** |
| function displayFacultyPage(){      deleteDisplayCurrentClass();      deleteDisplayCurrentProf();      deleteAll();      let bod = document.getElementById("container");      //Start of flex-container      let flexContainer = document.createElement('div');          flexContainer.className = "flex-container";          let pageTitle = document.createElement('h1');              pageTitle.id = "pageTitle";              pageTitle.appendChild(document.createTextNode("Fresno State - Computer Science Faculty"));          flexContainer.appendChild(pageTitle);          let facultyTable = document.createElement('table');              facultyTable.id = "faculty";              let thead = document.createElement('thead');                  let th1 = document.createElement('th');                      th1.appendChild(document.createTextNode("Instructor"));                  let th2 = document.createElement('th');                      th2.appendChild(document.createTextNode("Professor Name"));                  let th3 = document.createElement('th');                      th3.appendChild(document.createTextNode("Faculty Title"));                  let th4 = document.createElement('th');                      th4.appendChild(document.createTextNode("Contact Information"));              thead.appendChild(th1);              thead.appendChild(th2);              thead.appendChild(th3);              thead.appendChild(th4);          facultyTable.appendChild(thead);          for(let i = 0; i < nameProfArr.length; i++){              let tr = document.createElement('tr');                  tr.onclick = function (){                      profIndex = i;                      loadProfTemplate();                  };                  let tdImg = document.createElement('td');                      if(csciProfImages[i] == "N/A"){                          tdImg.appendChild(document.createTextNode("Image N/A"))                          tdImg.style.paddingTop = "95px";                          tdImg.style.paddingBottom = "95px";                      }                      else{                          let tempImg = document.createElement('img');                          tempImg.src = csciProfImages[i];                          tempImg.alt = "Statistics for player " + i+1;                          tempImg.style.width = "150px";                          tempImg.style.height = "180px";                          tdImg.appendChild(tempImg);                      }                  let tdName = document.createElement('td');                      tdName.appendChild(document.createTextNode(nameProfArr[i]));                  let tdFacultyRole = document.createElement('td');                      tdFacultyRole.appendChild(document.createTextNode(roles[i]));                  let tdContact = document.createElement('td');                      tdContact.innerHTML = emailAdd[i] + "<br>" + officeLoc[i] + "<br>";              tr.appendChild(tdImg);              tr.appendChild(tdName);              tr.appendChild(tdFacultyRole);              tr.appendChild(tdContact);              facultyTable.appendChild(tr);          }          bod.appendChild(facultyTable);  } |
| deleteScrollLinks() | This is used inside of the Department’s tab to delete the existing data inside of the tab. |
| **Program Description** |
| function deleteScrollLinks(){      let bod = document.getElementById("tabDepartmentContent");      if(typeof(bod) != 'undefined' && bod != null){          bod.innerHTML = '';      }  } |
| loadDepartmentAtoZ() | This loads the departments page onto the home page. It displays an A-Z page that allows users to choose from the departments in alphabetical order. This will help load the possible lists of classes that are inside of the database. In this version, only the Computer Science department and all its classes are available. |
| **Program Description** |
| function loadDepartmentAtoZ(){      deleteDisplayCurrentClass();      deleteDisplayCurrentProf();      deleteScrollLinks();      deleteAll();      let bod = document.getElementById("container");      //Start of flex-container      let flexContainer = document.createElement('div');          flexContainer.className = "flex-containerAtoZ";          let flexTab = document.createElement('div');              flexTab.className = "flexTab";          for(let i = 0; i < 26; i++){              let newBtn = document.createElement('button');              newBtn.className = "scrollLinks";              newBtn.onclick = function(){                  departMentPageBody.innerHTML = "";                      for(let j = 0; j < departmentNames[0][String.fromCharCode(i+65)].length; j++){                          let dep = document.createElement('h3');                          dep.appendChild(document.createTextNode(departmentNames[0][String.fromCharCode(i+65)][j]));                          departMentPageBody.appendChild(dep);                          if(String.fromCharCode(i+65) == 'C'){                              for(let k = 0; k < data[0]["rows"].length; k++){                                  let p = document.createElement('p');                                  p.appendChild(document.createTextNode(data[0]["rows"][k][0]));                                  p.onclick = function () {                                      for (let j = 0; j < data[0]["rows"].length; j++) {                                          if (p.innerHTML == data[0]["rows"][j][0]) {                                              classIndex = j;                                              displayClassInfoClass();                                              loadClassTemplate();                                              break;                                          }                                      }                                  };                                  departMentPageBody.appendChild(p);                              }                          }                      }                  flexContainer.appendChild(departMentPageBody);              };              newBtn.appendChild(document.createTextNode(String.fromCharCode(i+65)));              flexTab.appendChild(newBtn);          }          departMentPageBody.innerHTML = "";          for(let j = 0; j < departmentNames[0]['A'].length; j++){              let dep = document.createElement('h3');              dep.appendChild(document.createTextNode(departmentNames[0]['A'][j]));              departMentPageBody.appendChild(dep);          }          flexContainer.appendChild(flexTab);          flexContainer.appendChild(departMentPageBody);      bod.appendChild(flexContainer);  } |

**4.1.3 Database Access**

|  |  |
| --- | --- |
| **Class Name:** functions.js | |
| **Description:** This class is used to access the information inside the database. These will be modified in later versions to access more data inside of the databases | |
| **Function** | **Function Description** |
| getCsciName(index)  getUnits(index)  getOfferedIn(index)  getAdditionalInfo(index)  getDescription(index)  getCsciSyllabus(index) | All of these are basic get functions that allow “scrips.js” to access all of the class data that is inside of the database. It takes an index and gets all of the data to be presented on that specific class at that index. |
| **Program Description** |
| function getCsciName(index){      return data[0]["rows"][index][0];  }  // Gathers units from data file  function getUnits(index){      return data[0]["rows"][index][1];  }  // Gathers the semester they are usually offered in  function getOfferedIn(index){      return data[0]["rows"][index][2];  }  // Gathers any additional information related to the course  function getAdditionalInfo(index){      return data[0]["rows"][index][3];  }  // Gathers the description of the class  function getDescription(index){      return data[0]["rows"][index][4];  }  function getCsciSyllabus(index){      return data[0]["rows"][index][5];  } |
| getProfName(index)  getProfRole(index)  getProfDeskPhone(index)  getProfDepartMentPhone(index)  getProfEmail(index)  getProfOfficeLoc(index)  getProfDepartment(index)  getProfMailStop(index)  getCsciProfImages(index)  getCsciList(index)  getCsciLinks(index) | All of these get functions return the data for a professor at a given index from the database. This currently only works for the Computer Science faculty as the database only has their information in this current version. The index will correspond to the professor at that specific index and will give the information about them based on the given index. |
| **Program Description** |
| function getProfName(index){      return nameProfArr[index];  }  function getProfRole(index){      return roles[index];  }  function getProfDeskPhone(index){      return deskPhone[index];  }  function getProfDepartMentPhone(index){      return departmentPhone[index];  }  function getProfEmail(index){      return emailAdd[index];  }  function getProfOfficeLoc(index){      return officeLoc[index];  }  function getProfDepartment(index){      return department[index];  }  function getProfMailStop(index){      return mailStop[index];  }  function getCsciProfImages(index){      return csciProfImages[index];  }  function getCsciList(index){      return classList[index];  }  function getCsciLinks(index){      return additionalLinks[index];  } |

**5.** **Dynamic Model**

**5.1 Sequence Diagrams**

**5.1.1 Departments Page**

This sequence diagram shows how the Departments Page is loaded onto the home page.

**Diagram

Description automatically generated**

**5.1.2 Departments Page to Class Page Navigation**

This sequence diagram shows how users can navigate to the Class Pages through the Departments Page. It will display a single class’s data.

**Diagram

Description automatically generated**

**5.1.2 Faculty Page Navigation**

This sequence diagram shows how the Faculty Page is loaded onto the Home Page.

**Diagram

Description automatically generated**

**5.1.3 Faculty Page to Professor Page Navigation**

This sequence diagram shows how the Professor Pages can be navigated from the Faculty Page. The Professor Page will then be loaded onto the Home Page.

**Diagram

Description automatically generated**

**5.1.4 Search Bar Navigation**

This sequence diagram shows how the Search Bar can be used to navigate through this website. It is able to display both the Class and Professor Pages.

**Diagram

Description automatically generated**

**5.2 State Diagrams**

**5.2.1 Generate Class Page Using Department Page Navigation**

**Diagram

Description automatically generated**

**5.2.2 Generate Professor Page Using Faculty Page Navigation**

**Diagram

Description automatically generated**

**5.2.3 Generate Class or Professor Page Using Search Bar Navigation**

**Diagram

Description automatically generated**

**6.** **Non-Functional Requirements**

**6.1 Performance Requirements**

* The system should be able to generate the pages within 10 seconds of the user request.
* The system should allow multiple users to be able to interact with the system without error.
* The system should be able to search and retrieve data quickly.
* The available syllabus and links should work if they are provided for the class or professor.
* The database that is provided must be up to date and working while the server is being ran.

**6.2 Design Constraints**

* There is a limited amount of data that this website can provide.
  + Not all the data for each class and professor is available for this website to hold.