### CPSC S115: Introduction to Full-Stack Web Development

#### Prework & Module A

**Programming Fundamentals Ruby Basics and Object Oriented Programming Syllabus**

<table>
<thead>
<tr>
<th>Unit</th>
<th>Learning Objectives</th>
<th>Reference Topics</th>
<th>Competency Based Assignments</th>
<th>Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-Work</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 1 | • Understand the value and uses of version control with Git  
• Complete the full Git project flow  
• Work with the computer through the command line and BASH | • Git  
• commit  
• push  
• pull  
• remotes  
• add  
• fork  
• merge  
• BASH  
• intro  
• ls  
• cd  
• pipes | • Quiz: Git Basics  
• Git Todo  
• Git Flow  
• Git Merge Conflicts  
• Command Line Mystery | 15 |
| 2 | • Create basic web sites using HTML  
• Publish sites to different web hosts  
• Understand the distinction between styling and content | • Basic HTML  
• Lists  
• Tables  
• Images  
• Links  
• Validation  
• Site Publishing  
• Publishing to FTP  
• Publishing to Github Pages  
• HTML5  
• Forms  
• Media | • HTML Album Cover  
• HTML Blog  
• HTML Issue Bot 9000  
• HTML Map  
• Contact Forms  
• Code Along  
• HTML5 Video Codealong  
• HTML5 Semantic Elements  
• Codealong | 16 |
| | | | | |
| 3 | • Understand how to use CSS to style HTML elements  
• Lay block objects out using the box model  
• Create interfaces for mobile and web | • CSS Selectors  
• Class  
• Id  
• Element type  
• Page Layout | • CSS Kitten Wheelbarrow  
• CSS Graffiti Override  
• CSS Issue Bot 9000 | 8 |
<table>
<thead>
<tr>
<th>Page</th>
<th>topics</th>
</tr>
</thead>
</table>
| 4    | - Use the Bootstrap framework to increase speed of development  
- Box model  
- Overflow  
- Fluid Height  
- Float  
- Centering  
- Creating Columns Codealong  
- CSS Layout Quiz  
- ZHW Shoes Layout  
- Box Styles Codealong  
- Adding Responsive Features Codealong  
- Bootstrap Carousel Codealong  
- Codealong  
- CSS Layout Quiz  
- ZHW Shoes Layout  
- Box Styles Codealong  
- Adding Responsive Features Codealong  
- Bootstrap Carousel Codealong  |
| 5    | - Conditionals  
  - if  
  - switch/case  
  - Looping  
  - while  
  - for  
  - do...while  
  - Command Line Input  
  - gets  
  - Manipulating strings  
  - reverse  
  - upcase  
  - interpolation  
  - gsub  
  - Codealong  
  - CSS Layout Quiz  
  - ZHW Shoes Layout  
  - Box Styles Codealong  
  - Adding Responsive Features Codealong  
  - Bootstrap Carousel Codealong  |
| 6    | - Adding  
  - Push  
  - <<  
  - Removing  
  - pop  
  - shift  
  - Introspection  
  - flatten  
  - uniq  
  - include  
  - indexed access  
  - Iteration  
  - each  
  - map  
  - select  
  - detect  
  - Nested Data  
  - 2D arrays  
  - multi-dimensional arrays  
  - Hashketball  
  - NYC Pigeon Organizer  |
|      | - Why and when to use basic decision trees  
- Receive input from the Command Line  
- Manipulate strings  
- Work with Basic Looping  |
|      | - Why and when to use arrays  
- Work effectively with adding and removing from arrays  |
|      | - Acting on arrays with iteration and introspection  
- Retrieve, add and delete arrays nested inside other arrays  |
|      | - Why and When to use Hashes  
- Adding, removing and introspecting hashes  |
| 20   | - More Vowels  
- Badges and Schedules  
- Deli Counter  
- Phone Number Formatter  
- Cartoon Collections  
- Jukebox CLI  |
| 20   | - My Bach  
- Oxford Comma  
- Apples and  |
| 25   | - Holidays  
- Song Sorter  
- Tweet Shortener  
- Key for min value  
- Spotify API Lab  |
|      | - Hashketball  
- NYC Pigeon Organizer  |
<table>
<thead>
<tr>
<th>Module A</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
</table>
| • Manipulating and normalizing data into nested hashes | • Removing and introspection  
  • select  
  • map  
  • detect  
  • Nested Data | • Apples and Holidays |
| **Module A** | • Understand why Object Orientation was created  
  • Understand the benefits of Object Oriented Programming  
  • Differentiate between Instances and Classes  
  • Implement Methods and Properties in Classes | • History of OO  
  • OO Patterns  
  • Separation of Concerns  
  • Managing Internal State  
  • Classes vs. Instances  
  • Classes have behaviors  
  • Methods  
  • Classes have attributes | • Class and Instances Lab  
  • Instance Methods Lab  
  • Instance Variables Lab  
  • Object Attributes Lab  
  • OO Meowing Cat |
| 7 | **Instance Variables**  
  • attr_accessor  
  • setters  
  • getters | **Creating an init method**  
  • setting Instance Variables in init  
  • Other Defaults | **Object Initialize Lab**  
  **OO Basics** |
| 8 | **Understanding the Object Lifecycle**  
  **Create an initializer** | | |
| 9 | **Work with the concept of self**  
  **Understand Class level constructs** | **Class Constants**  
  • Creation  
  • Why do we use them?  
  **Self**  
  **Class Variables**  
  • Why use them?  
  • Using as a collection of all created instance s  
  **Class Methods**  
  • understanding the all method  
  • Creating class methods  
  **Private Methods**  
  • OO Programming as interface based program mining  
  • How to define | **OO Basics w/ Class Constants**  
  **OO School Domain**  
  **OO Counting Sentences**  
  **Class Variables and Methods Lab**  
  **Ruby Puppy**  
  **Advanced Class Methods Lab** |
| 10 | **Be able to model a domain with OO principles**  
  **Work with larger projects and chunk them into smaller segments** | **Domain Modeling**  
  • Deciding on properties  
  • Deciding on methods  
  **access levels** | **OO Email Parser**  
  **Anagram detector**  
  **OO Cash Register** |
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| 11 | Understand the power of relating objects  
Use a belongs to relationship  
Use a has many relationship  
Collaborate between two objects  
Use a has many through relationship |
|   | Why Relationships are important  
Belongs To  
Adding to all class variable  
Domain modeling a belongs to  
Has Many  
Reciprocal relationships  
Domain modeling a belongs to  
Adding to class variables in both directions  
Collaborating Objects  
Relation |
|   | Belongs to Object Lab  
Has Many Objects Lab  
Collaborating Objects Lab  
OO My Pets  
OO Kickstarter  
OO Banking  
Has Many Objects Through Lab |
| 12 | Understand the value of mass assignment  
Use simple meta programming |
|   | Mass Assignment  
Problem with manual assignment  
Passing in an options hash  
Meta programming  
Using the send method  
Using send to implement mass assignment |
|   | Mass Assignment Lab  
Custom Errors  
OO Triangle |
| 13 | Grab data from websites using Scraping  
Integrate third party code using gems  
Gem version control with Bundler |
|   | Gems and Bundler  
Where to find gems  
Open Source Licensing  
How to choose gems  
Installing and requiring gems  
Using a Gemfile  
Scraping  
Nokogiri |
|   | Scraping Flatiron  
Scraping Kickstarter |
<table>
<thead>
<tr>
<th>Unit</th>
<th>Learning Objectives</th>
<th>Reference Topics</th>
<th>Competency Based Assignments</th>
<th>Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>• Understand the uses of SQL</td>
<td>• SQL Introduction</td>
<td>• Bear Organizer</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>• Understand the layout of basic databases</td>
<td>• Installing SQLite</td>
<td>• Aggregate Functions Lab</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Install and run basic commands with SQLite</td>
<td>• SQL Basics</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Use Aggregate functions</td>
<td>• SELECT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• INSERT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• UPDATE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• CREATE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ALTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Using .sql files</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SQLite data types</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Aggregate Functions</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SUM</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• COUNT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• MAX</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• MIN</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Using .sql files</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>• Understand what a JOIN is</td>
<td>• Relational Databases</td>
<td>• SQL Crowdfunding Lab</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>• Understand foreign keys in has many/belongs to relationships</td>
<td>• History</td>
<td>• SQL Library Lab</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Understand Complex joins and Join tables</td>
<td>• Relations as related to ruby object relations</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• JOIN Statements</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• foreign keys</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Left Outer Join</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Left Inner Join</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Sorting and Grouping Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ORDER BY</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• GROUP BY</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Complex Joins</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Many-to-Man y</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Join tables</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SQLite gem</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Execute</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• create table</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• results</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 4 | - Understand the value and power of ORMs  
- build a simple ORM  
- Work with the inheritance and meta-programming required for ORMs | - Mapping Objects to tables  
- Dynamic create attributes  
- all method  
- find methods  
- Bi-Directional mapping  
- UPDATE  
- save checking for persistence  
- Dynamic ORMs  
- meta programmin g attributes  
- Refactoring common behavior to super | - Mapping to Table Lab  
- Mapping DB to Ruby Lab  
- ORM update Lab  
- ORM Bringing it all together  
- Dynamic ORM Lab |
| --- | --- | --- | --- |
| 5 | - Use ActiveRecord as an ORM  
- Understand all the features ActiveRecord provides  
- Use Rake tasks to perform database migrations | - ORM To ActiveRecord  
- Translate previous custom ORM methods into ActiveRecord method  
- Use ActiveRecord ::Base | - Translating ORM to AR  
- ActiveRecord Create, Read, Update, Delete  
- Rake Lab  
- Writing Migrations  
- ActiveRecord Costume Store  
- Activerecord Lab |
| 6 | | - Rake  
- Create custom rake tasks  
- Explain Rakefile  
- Migrations  
- Explain value of migrations  
- Write change migrations  
- Run migrations without rake  
- Run migrations with rake | - Activerecord- TVland |
| 7 | - Understand the history and current status of HTTP  
- Work with the HTTP Request cycle in ruby and Rack  
- Understand Rack as HTTP Middleware | - ActiveRecord Macros  
- has many  
- belongs to  
- has many :through  
- migrations with relationships | - Rack Todo  
- Rack Lab |
<table>
<thead>
<tr>
<th>Page</th>
<th>Topics</th>
</tr>
</thead>
</table>
| 8    | - Implement the HTTP Request/Response Cycle with Sinatra  
      - Understand the value of MVC and how to implement it in Sinatra  
      - Render front end with ERB templating  
      - Sinatra Hello World  
      - Sinatra Basic Routes Lab  
      - Sinatra Basic |
| 9    | - Understand how to gather user input through forms  
      - Work with form data to create responses  
      - Create and parse nested forms to create multiple models in one submit  
      - Dynamic Routes  
      - Symbol routing syntax  
      - Basic Forms  
      - Create HTML Forms  
      - Post those forms to your application  
      - Parse form data to create model objects  
      - MVC with forms  
      - full flow from form to controller, to model creation  
      - Nested Forms  
      - Using the name attribute of input elements  
      - Discuss the value of a nested form  
      - use the nested params hash.  
      - Basic Sessions Lab  
      - Sinatra Logging in and logging out  
      - Sinatra Sessions |
| 10   | - Be able to persist small amounts of data to the browser with sessions  
      - Understand the basic flow of logging in/out  
      - Browser data  
      - What are sessions  
      - What are cookies  
      - Inspecting cookies and sessions in a browser  
      - Sessions in Sinatra  
      - Setting the sessions hash  
      - Authentication  
      - Explain registration  
      - Basic Sessions Lab  
      - Sinatra Logging in and logging out  
      - Sinatra Sessions |
<table>
<thead>
<tr>
<th>Page</th>
<th>Topics</th>
</tr>
</thead>
</table>
| 11   | • How to integrate active record into sinatra  
      • Using multiple controllers in sinatra  
      • ActiveRecord and Sinatra  
      • Migrations  
      • rake and sinatra  
      • REST  
      • mapping routes to activerecord actions  
      • choosing routes constructively  
      • Multiple Controllers  
      • mechanics of multiple controllers  
      • How controller choose their view |
|      | • RESTful routes lab  
      • Sinatra Messages  
      • Playlist  
      • NYC Sinatra |
| 12   | • ActiveRecord associations in Sinatra  
      • join tables  
      • mapping tables to controllers  
      • Sinatra to Rails  
      • Router  
      • Controllers  
      • Models  
      • Views  
      • Static Requests  
      • ERB rendering  
      • Rails generator file structure  
      • Rails Hello World Lab |
|      | • RESTful Routing  
      • method definitions as http verbs  
      • what are the different methods  
      • Dynamic Routing  
      • Similarities to Sinatra  
      • How dynamic routing fits into REST  
      • RESTful index action lab  
      • Dynamic Request Lab |
| 13   | • Understand Routing and REST  
      • Create routes that connect to the appropriate action  
      • RESTful Routing  
      • method definitions as http verbs  
      • what are the different methods  
      • Dynamic Routing  
      • Similarities to Sinatra  
      • How dynamic routing fits into REST |
|      | • Rails URL Helpers  
      • difference between path and route  
      • Using this with link_to  
      • Create Action  
      • manual form tags  
      • POST action  
      • Edit/Update Action  
      • difference between form and action  
      • Rails URL Helper Lab  
      • Rails Form Tag Lab  
      • Rails Create Action Lab  
      • Rails Index/Show/New/Create Lab  
      • Rails form_for Lab  
      • Formal Affair Rails  
      • Rails form_for Lab  
      • Rails URL Helper Lab  
      • Rails Form Tag Lab  
      • Rails Create Action Lab  
      • Rails Index/Show/New/Create Lab  
      • Rails form_for Lab  
      • Formal Affair Rails  
      • Rails form_for Lab |
| 14   | • Create dynamic views with ActionView  
      • Collect user input using form_for and rails scaffold  
      • Rails URL Helpers  
      • difference between path and route  
      • Using this with link_to  
      • Create Action  
      • manual form tags  
      • POST action  
      • Edit/Update Action  
      • difference between form and action  
      • Rails URL Helper Lab  
      • Rails Form Tag Lab  
      • Rails Create Action Lab  
      • Rails Index/Show/New/Create Lab  
      • Rails form_for Lab  
      • Formal Affair Rails  
      • Rails form_for Lab |
|      | • Form_for  
      • Rails Blog Scaffold |

- **Login creates a session**
- **logout destroys the session**
<table>
<thead>
<tr>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>• Understand the power of validations</strong></td>
<td><strong>• Use ActiveRecord in Rails</strong></td>
<td><strong>• Understand the power of refactoring to components</strong></td>
</tr>
<tr>
<td><strong>• Create custom validations</strong></td>
<td><strong>• Understand complex many to many through</strong></td>
<td><strong>• Use partials for collections, forms and other reusable views</strong></td>
</tr>
<tr>
<td><strong>• Work with built in validations</strong></td>
<td><strong>• Using AR to search and sort through data</strong></td>
<td><strong>• Use helpers to remove logic from views</strong></td>
</tr>
<tr>
<td><strong>• Display validation errors to users</strong></td>
<td><strong>• Associations</strong></td>
<td><strong>• Partial Collection</strong></td>
</tr>
<tr>
<td><strong>• Validation callbacks</strong></td>
<td><strong>• through</strong></td>
<td><strong>Partials</strong></td>
</tr>
<tr>
<td><strong>• presence</strong></td>
<td><strong>• Domain modeling</strong></td>
<td><strong>Partial Collection</strong></td>
</tr>
<tr>
<td><strong>• uniqueness</strong></td>
<td><strong>• Working with JOIN tables</strong></td>
<td><strong>Partial Form</strong></td>
</tr>
<tr>
<td><strong>• numeric</strong></td>
<td><strong>• Searching</strong></td>
<td><strong>Partial Cross Controller</strong></td>
</tr>
<tr>
<td><strong>• regex matching</strong></td>
<td><strong>• Where</strong></td>
<td><strong>Partial Helpers</strong></td>
</tr>
<tr>
<td><strong>• Custom validations</strong></td>
<td><strong>• Find</strong></td>
<td><strong>Helper Presenter Pattern</strong></td>
</tr>
<tr>
<td><strong>• Errors</strong></td>
<td><strong>• ordering</strong></td>
<td><strong>Helper Logic in Views</strong></td>
</tr>
<tr>
<td><strong>• The @errors variable</strong></td>
<td><strong>• AREL</strong></td>
<td><strong>Helper Cross controller helpers</strong></td>
</tr>
<tr>
<td><strong>• Describing errors in form_for</strong></td>
<td><strong>• Blog Associations Validations</strong></td>
<td><strong>Flatiron BNB Methods</strong></td>
</tr>
<tr>
<td><strong>• resolving errors</strong></td>
<td><strong>Pollywog Ranch</strong></td>
<td><strong>Programmer Profile Partials</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Simple Search</strong></td>
<td></td>
</tr>
</tbody>
</table>
Module C

Javascript: Creating Interactive and Performant Front Ends with Javascript Syllabus

<table>
<thead>
<tr>
<th>Unit</th>
<th>Learning Objectives</th>
<th>Reference Topics</th>
<th>Competency Based Assignments</th>
<th>Hours</th>
</tr>
</thead>
</table>
| 1    | - Understand the similarities between ruby and javascript  
      - Know when javascript is the correct language to use  
      - Use Javascript to do basic programming tasks | - JS Basics  
      - How to run JS code  
      - How to read JS tests  
      - JS Data Types  
      - Hashes  
      - Variables  
      - Functions  
      - Strings  
      - Math  
      - Object Literals  
      - JS Control Flow  
      - if statements  
      - looping | - JS Debugging  
      - JS isBetween  
      - FizzBuzz  
      - Donut Lab | 20 |
| 2    | - Use closures to modify variable scope  
      - Understand prototypical inheritance | - Closures  
      - Lexical scope  
      - Scoping with Var  
      - Anonymous Functions  
      - Inheritance  
      - Prototypes  
      - Constructors  
      - Classes as Hashes | - Space Invaders  
      - JS Tweet Shortener  
      - Triangle.js  
      - Robot Simulator  
      - JS Tic Tac Toe | 10 |
| 3    | - Understand when and how to use jQuery  
      - Work with other Libraries  
      - Use Selectors in JS | - Basic jQuery  
      - JS Libraries  
      - History of jQuery  
      - the $ selector  
      - CSS Selectors in JS  
      - Selectors  
      - iteration  
      - Convenience Methods | - jQuery Calculator  
      - jQuery OO Tic Tac Toe  
      - Task Lister  
      - Flickr Image Search | 10 |
| 4    | - Query and parse a JSON API  
      - Update on screen content dynamically | - AJAX  
      - JSON parsing  
      - Using $.ajax  
      - Async Javascript  
      - Displaying results  
      - JSON | - Temperature Visualizer  
      - Spotify with AJAX  
      - Inflation Calculator  
      - Cars Lazy Loading | 30 |
2Week Student Projects Syllabus

During this portion of the course students are tasked with working solo to create a project. This course provides an in-depth opportunity for student to demonstrate their learning accomplishments and get a feel for what working at a company is really like.

Students are divided randomly and begin with an initial pitch session. Students are tasked with pitching three different ideas that they are passionate about. Instructors take the pitches and then choose the final project that the students work on. Instructors choose projects based on difficulty and feasibility given the time constraints of the course. Specifically, the projects are required to have the following items to meet the minimum bar of difficulty:

- Must work directly with an API
- Must have > 3 Models and Controllers
- Must use helper methods to remove logic from views
- Must include at least two complex joins
- Must have some AJAX functionality

<table>
<thead>
<tr>
<th>Unit</th>
<th>Learning Objectives</th>
<th>InClass Lectures</th>
<th>Lab Activities</th>
</tr>
</thead>
</table>
| 1    | • Understand developer flow with GIT | • Git  
  • branches  
  • Feature Branches  
  • Advanced Merge Conflict  
  • GitHub  
  • Pull Requests  
  • Issues | none |
| 2    | • Break a large project into workable chunks | • Minimum Viable Products  
  • Choosing priority features  
  • User Stories  
  • Project Management  
  • Dividing work  
  • Agile Workflow | none |