Developer Guide

**Coding guidelines:**

* Uniform appearance guidelines amongst engineers with good annotation
* Website completed to W3C compliance, universal web standards that provide a set of guidelines for clear, concise code
  + High-efficiency code for fast page loading and easy navigation on older browsers
  + Standardized indenting and annotations
  + Device compatibility, allowing reformatting for different browsers and devices, including mobile
  + Search engine compatibility - readable through web crawler for accurate and quick recognition by search engine bots
  + Accessibility and disability standards for visually impaired through following accepted web standards governing semantic structure
  + Enforced through collaborative filtering of code, “code checkups” before submitting each sprint for cohesiveness and effectiveness.
* Code used will be in in HTML5, Javascript, CSS, PHP, XHTML
* W3 guideline example can be read on this page https://www.w3.org/standards/

**Software toolset:**

* MySQL - used for database development and refining
* Sublime text/JetBrains/Visual Studio - IDE/ text editors used for HTML / CSS writing and management, javascript integration
  + Emmet extension in VS for HTML/CSS Workflow
* Discord - for team communication/screen sharing, light file sharing
* GitHub - iterative development approach, file sharing, bundling files for client

**Software design:**

For the software designed together with the client, we decided to create a website where the user can see five different modules/pages. "Home, About Us, Contact Us, Login, Create Account." For the "Home" page, we will introduce the user to the overall idea of what the company is about. For the "About us,” we provide detailed information on what the company is about and its ideology. For "contact us," the user will have the opportunity to submit any question or concern that one might have. This information will be sent in an email to a specific company's email. "Log in" for the users that already created an account to apply for the job. "Create an account." For the time that a user decides to apply for an available position for the job.  At this moment, we are planning to have a database where we will be able to store the user's information from the c" create an account and apply for a job." Section.  The reason for making the database is to allow us to store the data for logistics purposes. Due to the simplicity of the project scope, parallel development strategies will not be used apart from the database to website integration phases in a testing atmosphere, otherwise, purely iterative development will be used.

**System components:**

1. **Front End component:**

* **Navigation:** help users browse resources by section, such as the main menu.
* **Mockup:** to separate the page elements (text container, buttons, links).

1. **Back end component:**

* **Contact form:** fill out the personal information**.**
* **Online databases:** operate with data online(search, delete, add).
* **Protected sections:** can be accessed by confirming the special access rights.

1. **Hosting:** where your website is physically located.

**Discussion of data storage:**

We are going to use MySQL as the programming language for our database. We are planning to use desktop storage, or maybe cloud storage. We are not sure yet, but we are going to discuss this topic at the next meeting.

**Alternative discussion:**

For the moment, we have not found or discussed any other architecture where the team has come up with a different alternative.

**What are the triggers and frequency for your automated builds and why?**

Our triggers for the automated build will trigger upon completion of each site element (apply page, contact us page, etc) for subversions, which will culminate into main versions for each sprint, in which all elements will be tested thoroughly for cohesion, security, and bugs. The interval for subversions should be about 72-hour spans, where full versions will likely be twice per sprint.

**What tests will be executed in a CI build?**

In this phase, we run automated tests to validate our code’s correctness and the behavior of our product. The test stage acts as a safety net that prevents easily reproducible bugs from reaching the end-users. Failure during the test stage exposes problems in code that developers didn’t foresee when writing the code. It’s essential for this stage to produce feedback to developers quickly. Through Selenium Client’s WebDriver API, we will use tailored test scripts to interact with page and application elements to simultaneous test various browsing conditions, with default test cases supplemented by community and custom made script. Using the selenium test automation framework, we can compare the results of performed test cases and make any necessary repairs to the website following the feedback. For our server, we can use the Selenium Server module, with WebDriver API, to run tests against web browsers on the server machine, to test for security and effectiveness using a variety of pre-built and custom tooled tests. Since we are supplementing Selenium with Cucumber, additional web support, as well as client to developer transparency, is allowed, creating a more efficient test bed with a clear cut development path.

**What development actions trigger a CI build?**

Failure in each stage typically triggers a notification—via email, Slack, etc.—to let the responsible developers know about the cause. Otherwise, the whole team receives a notification after each successful deployment to production. CI builds will typically be triggered upon completion and testing of developer responsibilities as noted by the pivotal tracker and cucumber software, which should be compiled into a newer version about every 72 hours at the current planned development pace.

**How to add new code to the test base?**

First, we initiate the WebDriver interface, which will be the first step in writing the selenium test for our project. Then we create an instance of the WebDriver interface utilizing a constructor from the web browser we are using. The constructor's name depends on the web browser, and utilizing those constructor depends on the programming language. Once we create the instance of the WebDriver interface, with the instances, we can invoke methods. To add a new code, we assign the instance to a variable when we create it, and we use the variable to invoke the methods in which those methods will be the new code or functions added to the test base.

**Your test automation infrastructures (e.g. Junit, Mocha, Pytest, etc.)**

Our test automation infrastructures will include Selenium paired with the Cucumber BDD for performing acceptance testing for web applications. Paired together, these programs allow for automation of functional validation in a combination of natural English as well as code framework. Cucumber’s main allure is the concept of feature files, which are used to create test automation steps/acceptance tests. Feature tests can provide important roadmap information for our client, who is unfamiliar with coding specifics, and the development team, who ideally can tackle small pieces of code and combine at certain sprint intervals. Through the webdriver interface paired with a supporting browser such as firefox, the webpage hosted from github can be easily instantiated and used to test the various programming languages such as javascript, HTML, PHP, and CSS used for our website.

**Your CI service and how your project repository is linked to it.**

Our CI service of choice is Travis, which will be linked to our project repository through GitHub. The database element can also be tested and temporarily hosted on Travis for iterated database testing.

Code Architecture

1. Menu Index
   1. Index.html - landing page, connects to all other site elements. Will receive increased detail and more available information from landing page
   2. Index.css
2. Secondary Layers
   1. createAccount.html – html framework for account creation
   2. createAccount.css – styling for account creation
   3. aboutUs.html – information page accessible from menu – should later have scrolling information settings/customer testimonials
   4. aboutUs.css – styling for about us page
   5. apply.html – application page, should have specific position application in future and direct email portal
   6. apply.css
3. Tertiary Layers
   1. Form.html – bus driver application form

Specialized Testing

* Security
  + “Vega” test software for SQL, XML security testing. Open source + free. To run, download software from <https://subgraph.com/vega/>
  + Vega is a free and open source web security scanner and web security testing platform to test the security of web applications. Vega can help you find and validate SQL Injection, Cross-Site Scripting (XSS), inadvertently disclosed sensitive information, and other vulnerabilities. It is written in Java, GUI based, and runs on Linux, OS X, and Windows.
* Browser Support
  + “Powermapper” website tools
* [OnDemand Suite](https://www.powermapper.com/products/ondemand/)
* [Desktop Suite](https://www.powermapper.com/products/desktop/)
* [SortSite Desktop](https://www.powermapper.com/products/sortsite/)
* [PowerMapper Desktop](https://www.powermapper.com/products/mapper/)

“Powermapper” will be used to check for browser support and device compatibility, which can detect

* CSS features not supported by some browsers
* HTML tags not supported by some browsers
* Vendor specific HTML and JavaScript
* Image formats not supported by all browsers (e.g. some transparent PNGs don't display correctly on old versions of Internet Explorer)
* Technologies not supported by some browsers (e.g. Flash on iPhone and BlackBerry)

**A combination of these tests should provide sufficient coverage for the intended scope of our product.**