NAME Divna Mijić Project 1 – Rubric

ISTE340 Client Programming

|  |  |  |
| --- | --- | --- |
| **✔** | **Tasks** | **Points** |
| **Web Design and Programming Principles** | |  |
| ☐ | Professional looking web application, following modern web design principles and practices, that provides a meaningful content to the user:   * Look & Feel (10 pts) * Content with minimum of 300 words (10 pts)   Using a CSS framework to implement the look and feel and the structure of the web app is welcoming. Note: This rubric requires knowledge from ISTE-140 Web & Mobile I AND ISTE-240 Web & Mobile II TODO: Explain what you did here, and if applicable, where to find this feature in your code (e.g. init.js, line12)  The website is professional-looking with its high-resolution photos, warm and compatible color palate and custom-made icons and imagery. Furthermore, it follows the CRAP practices as the color scheme is consistent throughout the website, content is not cramped and the employment of the same font and colors offer the user a sense of organization, unity, and consistency.  🡪 This can be found through the whole of CSS file, but some of the lines are for instance index.css 503-510 & index.css 378-387 | 20 |
| ☐ | Professionally structured & implemented website, following good programming principles and techniques (OO approach with the MVC architecture based on ES6 features)  TODO: Explain what you did here, and if applicable, where to find this feature in your code (e.g. init.js, line12)  The code is clean, readable and heavily commented. Furthermore, for instance: multiple methods are used for instead of a lot of code cramped into one function. This is especially true for the form validation which can be found on lines FormValidator.js line 18-89 | 10 |
| ☐ | The code is properly aligned and indented AND documented using comments in the expected format (JSDoc) that clearly explain the code functionality.  TODO: Explain what you did here, and if applicable, where to find this feature in your code (e.g. init.js, line12)  This is surely done as all the code is properly formatted and commented. One such example can be found on CakeView.js line 111-118 | 10 |
| **Conditional/Cascading Selects** | |  |
| ☐ | Each select must have at least 2 choices and there must be a depth of 3 different selects  TODO: Explain what you did here, and if applicable, where to find this feature in your code (e.g. init.js, line12)  The user has 3 choices they can choose from for their cake of choice; layers, flavor and decorations. The implementation of such a function can be found mostly in the CakeView.js | 5 |
| ☐ | Selects are dynamically created depending upon the user’s previous select option.  TODO: Explain what you did here, and if applicable, where to find this feature in your code (e.g. init.js, line12)  Each select does indeed have 2 distinct options depending on the previous (if applicable) choice made. The implementation of such a function can be found mostly in the CakeView.js | 5 |
| ☐ | The user can start over and the selects are re-drawn from the point of the user’s new selection  TODO: Explain what you did here, and if applicable, where to find this feature in your code (e.g. init.js, line12)  I have decided to implement a “restart” button that appears every time at least one selection has been made. This feature can be found in CakeController.js on line 111 | 5 |
| ☐ | The selection process includes a dynamically generated UI element that changes in response to the user's selection  TODO: Explain what you did here, and if applicable, where to find this feature in your code (e.g. init.js, line12)  The preview of the cake the client is in the process of or done creating is displayed on the right of the form. This can be found in the CakeView.js at line 91 as well as the index.html itself. | 10 |
| **Form Element** | |  |
| ☐ | Once the user is done with the selection, a form is used to send the user selected options as well as additional data that are application specific (e.g. email, age, ...).  TODO: Explain what you did here, and if applicable, where to find this feature in your code (e.g. init.js, line12)  The form got created, as well as text inputs expected from the user. Upon successful and valid input of user data, the cake object in the local storage is modified. This is done with the usage of multiple files, but can surely be found in the FormModel.js file. | 5 |
| ☐ | The form inputs are validated, and an informative feedback with restyled form elements is provided in case of invalid data. At least 2 inputs with an expected format to validate are provided. Validation needs to be performed by JavaScript not by HTML5.  TODO: Explain what you did here, and if applicable, where to find this feature in your code (e.g. init.js, line12)  Validation is set on the phone and email inputs. Upon every non-valid entry, a text message is displayed and submission not possible. This is also done using more than one file, but some code can be found in FormController.js | 10 |
| **Local Storage** | |  |
| ☐ | Once the user is done with the selection, Local Storage is used to store the selected options.  TODO: Explain what you did here, and if applicable, where to find this feature in your code (e.g. init.js, line12)  Upon successful selection, the “cake” object is stored and can be viewed in the local storage. | 5 |
| ☐ | The data regarding the user choices are loaded from Local Storage into the form. On form submit, all the data are stored in Local Storage  TODO: Explain what you did here, and if applicable, where to find this feature in your code (e.g. init.js, line12)  Upon successful selection, the “cake” object, alongside valid user input as new attributes of the object cake are stored and can be viewed in the local storage. | 5 |
| **Browser Support** | |  |
| ☐ | The application checks how well the user’s browser supports the application’s features and lets the user what they need to do to have the application work as expected. The check is performed in the beginning of the program, before allowing the user to interact with the web page.  TODO: Explain what you did here, and if applicable, where to find this feature in your code (e.g. init.js, line12)  Modernizr was used as well as a container that will be displayed in cases of different browsers not supporting some of the JS as well as CSS features. This is the reason modernizr-custom.js can be found inside the vendor folder in the workspace | 10 |
| **TOTAL:** | | **100** |