CSC 322 FINAL REPORT

Group R: Chad-Good Politziyia Tech

Item 1: 100%

^ Personalized html pages for each userType.

Why? The color of the navBar and description about userType and userName within the navBar changes based on the userType signed in. In addition, the log-in button changes to log-out if a userType is greater than a Visitor; This also applies to home.html, where the button "Register for Customer" changes to "View Account" if a userType greater than Visitor is logged on. For userTypes "Owner" and "Employee", the default HTML document displayed is "admin.html", while userTypes "Customer" and "Visitor" are shown "home.html" when first logging on.

Item 2: (0.5 + 0 + .5 + 1)/4 = 50%

^ Employees set up custom builds. (Back-end only 50%).

^ PC's tailored towards different uses/buckets (0%)

^ Customers can also make custom builds (Back-end only 50%)

^ Custom build checks for hardware constraints (Back-End by default, thus 100% implemented in project)

Item 3: (0.66 + 1 + 0.8 + 0. 5 + 0.1) / 5 = 61.2%

^ Freely browse the available items (Users can browse the available items, but clicking on them doesn't work so details are not shown on front end, but available in back-end: (100% + 0% + 100%) / 3 = 66%)

^ Visitor can apply to be a customer by signing up with personal information (100%).

^ Employee processes information (Website sends forum to employee stating that a visitor applied for customer and requires action: 100%; However, the button to approve or deny the application wasn't fully integrated by the time the demo was presented, and only missing this small step: We argue for 80% since the biggest hurdle behind the whole process for getting employee action is mostly backend and fully coded out, however the button to call the backend function was the only thing missing)

^ Rejected memo by Employee sends to owner (The process to send the rejected memo to the owner exists in the backend, but wasn't implemented because the front-end couldn't approve, and thus disapprove was not implemented either: 50%)

^ Owner can veto disapproved application or agree with the Employee's decision (Veto function is in the website's code, however it relied on helper functions that couldn't be implemented due to time constraint, however their behaviors were listed out in previous reports. Lastly, the owner can't call on front end because the button still doesn't exist [Though this kind of feels like double jeopardy]: 10%)

Item 4: 100%

^ Customer can deposit money (Shown in Demo! 100%)

^ Customer gets warning when trying to purchase with insufficient funds (Shown in Demo! 100%)

Item 5: 75%

^ Customer making custom build checks compatibility (Solely backend: 100%)

^ Prompted choice to push custom into catalog (The choice to push the custom build into the catalog doesn't exist, however the back-end function to push a custom build PC into the catalog does exist. We believe 50%)

Item 6: (37.5 + 0 + 0 + 66% + 16.67%) / 5 = 24.167%

^ After purchase Customer can rate the configuration (Function to push rating exists in backend; No function/button to rate a device exists on the front-end; Database however stores information about previously bought builds which would've been the check to allow customer to rate: We argue 50% [back-end] + 0 [no button] + 0 [no front end function] + 100% [storing info as a check] = 37.5%)

^ >= 3 five-stars $\wedge$ 0 one-stars → [Compliment User] (Not implemented; 0%)

^ >= 3 one-stars $\wedge$ 0 five-stars → [Warn User] (Not implemented; 0%)

^ Customer can communicate with store employees about purchase, etc (HTML pages for Contact Employee and Employee forum to receive notification exists; The front end button to instigate the back-end act of pushing a forum for communication does not exist, however the back-end contains the proper functions to instigate this communication and to notify the Employee; Employee cannot respond to instigation as stated in Item 3 [We claim Double Jeopardy! If this button existed the Employee would've been able to respond]: 50% (HTML for customer instigation, however button to call function does not) + 100% (Employee forum HTML exists and shows requests, and we claim double jeopardy for pressing button to respond to request) + 50% (Functions in back-end to carry out and store customer request and employee response exists, however the main function call in the JS has not been completed): 66%)

^ Customer and Employee can complain or compliment about the other party, which is sent to the Owner (Back-end to send complaint/compliment to Owner exists and can be stored, but Owner cannot respond as stated previously (not double jeopardy since it's a different specification); Customers and Employees are not prompted to send review at all in our code. Lastly, no function was developed to allow the owner to determine how many compliments/warnings to give based on detailed behavior: Thus 50% + 0% + 0% = 16.67%)

Item 7: (50% + 50% + 0% + 33%) / 4 = 33.3%

^ Customer receiving up to 3 warnings get kicked out of the system (DB is organized to recognize that a Customer Account should be disable after given 3 warnings, however no update function was implemented to check and disable the account: 50%)

^ Same for Employees receiving Demotion for 3 warnings, and Promotion for 3 compliments (50%)

^ However, it was not implemented in schema to recognized Employees getting fired for 2 Demotions (0%)

^ But for Customer receiving 3 compliments, the system also recognizes that the customer should receive a 10% off discount, but the functions to update the DB and to apply the 10% off to the purchase have not been implemented (33%).

Item 8: 16.66%

^ Worst rated PC's are removed and person gets warning (Back-end function to remove PC exists, but eventlistener to detect this change in the DB and push warning on user are not implemented: 33%)

^ Best rated PC's are pushed to top and the one who set it up received a compliment (Back-end function to push PC to top was not actually implemented at all because the local database is an unsorted dictionary and thus wouldn't establish a hierarchy for the PC's. This could've been fixed by adding another attribute for the pc_build dictionary schema, but we must've forgotten to do that: 0%)

Item 9: (50% + 66% + 0% + 0%) = 29.167%

^ Comments are compared to dictionary of bad words to check for foul language (dictionary exists, but parsing of comments function does not to compare to said dictionary: 50%)

^ Comments with bad words by visitors are removed: (Back-End to recognize if the comment is by a visitor exists [check if commentor==”NoName”]. Back-End to remove comment from Device DB schema [where the comments are held] also exists. However javascript EventListener to fire when a comment is posted and to check if a comment should be removed does not exist: 100% + 100% + 0% = 66%.]

^ Comments with bad words by customers are replaced with correct amount of * characters. (not implemented 0%).

^ Accounts (not visitor) that comment with 1-2 bad words get 1 warning, 3 or more gets 2 warnings (not implemented: 0%)

Item 10: 90%

^ Creative Feature Accounting for 10% of the system (To improve security of the website we as a group unanimously decided to hold all navigation of the website within a single iframe. This prevents visitors and customers from accessing files through direct URL. It's not entirely flawless, and still vulnerable to attacks, however through mindless clicking of buttons, the user will never access pages they're not supposed to. Since the entire navigation is done through the iFrame, we argue this is clearly more than a 10% creative feature since we had to adapt the entire system to work with this framework. To instigate this feature, the program must be launched through launch.html, which wasn't displayed during the demo since it wasn't entirely working at the time, and the fix was only found after the fact. Due to the tardiness, we think a 10% deduction is within reason: 90%)

Overall Grade:

(100% + 50% + 61.2% + 100% + 75% + 24.167% + 33.3% + 16.66% + 29.167% +90%) / 10 = 57.9495… ≈ **58%**

Thus, we unanimously agree that we do not deserve a passing grade for this project, however a 58% correctly reflect all of the effort we have conducted thus far. Since one of our members avoided contact during the last 3 weeks only to drop the course at the last minute, the group was unsure to go out of their way to makeup their assigned work for we didn't want to steal points away from them if they managed to pull through at the last minute. In addition, Alex was a bit slow with his work and the entire group wants to make this clear to the professor as well. We know that Alex put his best work forward and deserves no deduction for his conduct, however this disabled the group's ability to proceed to higher-up functions due to slow development on his end.

We suggest 1 of 2 options. Either we should grade everyone out of 80% instead of 100% due to the 20% work allocation being lost from the group member dropping; Or Option 2 being the simple case of: if the group member did not drop the course, we would've equally unanimously agreed that they didn't contribute toward the demo, and would have his 58% grade shared equally among our teammates.

AR: 100%
GO: 100%
IA: 90%
IC: 95%
Ratul: 5% (Helped with Phase 1 report, on 2 cases, though they promised to contribute more after the fact)