**Team Name:** Color Laville

**Team Players:** Taylor Laing and Collin Neville

**Project Title:** Covert Tees

**Application URL: https://covert-tees.herokuapp.com/**

**Project Summary:**

Covert tees is a business that sells t-shirts out of a physical location and has been doing so for years. However, due to recent shifts in the economy, Covert Tees understands they will have to expand their operations to serve the online market. Doing this presents some problems: instead of written transactions, all orders will need to be accessed online.

The purpose of our application is to create an online store where users can shop for t-shirts. They can create an account, view products, login, add an item to the cart, and checkout. For non-client users (admins), the purpose is to be able to edit product categories, images, and view a list of all users. Admins get special access to certain pages in order to make changes to the database.

Some features include a contact us page, adding a product to the cart, and a login page for users.

**Routing Guide:**

GET /account finds user with current ID and renders the account/index view

GET /account/logout logout and redirect to /account

DELETE /account/delete remove Account with userID and redirect to an admin users view

POST /account/update update account from ID and redirect to /account

POST /account/update/shipping update currentUser shipping\_address and redirect /account

POST /account/create create an account and redirects to home page

GET /admin calls getAll on every controller and renders an admin/index view

GET /admin/product/create gets all data from categories and variations and render admin/product/index view

GET /admin/product/:slug gets product, category, and variation from slug ID and renders admin/product/index view

GET /admin/category/create renders shop/category view

GET /admin/category/:slug indexes Category by slug and renders shop/category view

GET /admin/variation/create get all variation types and render admin/variation/index view

GET /admin/variation/:slug indexes Variation by slug and renders admin/variation/index view

GET /admin/user/:id/remove deletes User and redirects to an admin view page

GET /admin/user/:id/:admin/toggle-admin toggles admin for a User and redirects to an admin user age

GET /admin/image/:id findbyID an Image and then redirects to admin/image/index view

POST /cart/add add item to cart and save cart

POST /cart/update update quantity in cart and save cart

POST /cart/remove/:id remove item from cart and save cart

GET /cart/checkout ensure user is logged in and render cart view

POST /cart/stripecheckout saves order and uses stripe to checkout

GET /cart/confirm confirm payment with stripe, and reset cart, and render invoice view

GET /cart render cart/view view with cart information

POST /category/update update category by ID and redirect to redirect to admin category view

POST /category/create create a category, save it, and redirect to admin/category view

POST /category/:slug/remove remove category

GET /contact: render contact/index view

POST /contact/submit creates new contact, saves it, and redirects to /contact

GET / render /index view

GET /about render /about view

GET /login render account/login view

POST /login authenticate account

POST /image/upload creates a new Image instance and saves it

POST /image/update finds image by ID and updates it, then redirects to an admin page

POST /image/:id/remove updates Product and User to delete images, then redirects to an admin image view

GET /order render order/index view

POST /product/update update Product by ID and redirect to an admin product view

POST /product/create creates a Product instance and saves it to the database. Redirects to an admin product view

GET /product/:slug retrieve product by slug

POST /product/:slug/add-image add image to product

POST /product/:slug/remove remove product

GET /product/:slug/image/:imgID/remove remove image from product images

GET /shop get all categories and products and render shop/index

GET /shop/:slug gets category by slug and renders shop/index

POST /subscriber/subscribe creates new Subscriber with user email and redirects to the home page

POST /variation/create creates a new Variation from a form and redirects to an admin page view

POST /variation/update updates Variation by ID and redirects to an admin page view

GET /variation/:slug finds one Variation from slug value and stores variation in locals.

POST /variation/:slug/remove finds a Variation from slug value and deletes the Variation. Redirects to /admin

**User Credentials and Roles:**

**Administrators:**

**Exclusive Route Accounts:**

/admin

/admin/product/create

/admin/product/:slug

/admin/category/create

/admin.category/:slug

/admin/variation/create

/admin/variation/:slug

/admin/user/:id/remove

/admin/user/:id/:admin/toggle-admin

/admin/image/:id

/image/upload:

/image/update:

/image/:id/remove:

***Sample user account:***

admin@user.com

Admin$321

**Bug Hunt Response:**

|  |  |
| --- | --- |
| **Comment** | **Response** |
| All of the routes work as described and seem to function properly, except for the subscriber/subscribe route. | This was resolved during the bug hunt: subscriber/subscribe is functioning and can be found in the footer of the website (the team was unable to locate it and left this comment in response). |
| The one thing we had a hard time finding was the subscribe button. | Subscribe feature was designed small and out of the way in the footer as it is not a prominent feature to be used within the site. |

**Project Requirements:**

|  |  |
| --- | --- |
| 1. 5 interlinked *dynamic*  pages (views) that display dynamic, database-driven content **per team member**. | Views that display dynamic content from database and associated queries/transactions:   1. /  Queries and displays latest products and categories, and header displays user name and cart count 2. /shop Gets all categories, and through a hash url we created subpages that are also dynamic based on each category available (4 total pages given current 4 categories) 3. /product/:slug Gets all information pertaining to a product, including images, title, category, description, variations, and the ability to add the product to the cart 4. /account Gets user information and displays it to be consumed or updated by the user (including image, name, email, orders, and shipping address). There are 3 dynamic views using a hash url creating 3 dynamic sub pages (account details, orders, shipping address). 5. /admin received all information about orders, products, categories, variations, images, contacts, subscribers, and users. This was done to create a fluid user experience while navigating pages that are now preloaded. This accounts for 8 dynamic pages each accessed via hash url. 6. /order/:id receives order information and displays the associated invoice for that order 7. /admin/product/:slug receives product information to be updated as needed (including details and images, each with a hash url). 8. /admin/category/:slug receives category information to be updated as needed (including details and images). 9. /admin/variation/:slug receives variation information to be updated as needed 10. /admin/image/:id receives image information to be edited as needed 11. /cart displays dynamic information based on items the user adds to their cart |
| 2. At least 5 database transactions (a transaction includes selecting, inserting, updating, or deleting data from the database) **per team member** | See above for information on routes and pages that utilize the numerous functions available. All items can be created, updated, or deleted and, as such, viewed on user interface. |
| 3. Support for user authentication and secure storage of user credentials | User credentials are managed through the Passport.js package, and users authenticate through a local authentication strategy that verifies the user credentials through the user mongoose model. |
| 4. Preservation of state (may include cookies, session state, url parameters, querystring, etc.) | The session stores information about the currently logged in user and their permissions, as well does it hold the cart and its data (as it is not yet an order). Slugs are used as url parameters when accessing specific pages, such as product pages |
| 5. Development of at least one REST API and consumption of another third-party API | Our REST API: Log in using /api/login and entering email and password as json body data, then use the returned token as a header parameter in the call to /api/products .  Third Party API: Retrieving the weather for Logan, Utah and displaying it in the top menu bar. |
| 6. Inclusion and use of at least two npm libraries that are *not*  covered elsewhere in the course (i.e., you need to explore additional functionality on your own). | To write files such as images, the following were used: path, express-fileupload, and fs. |
| 7. Input validation and error handling for all free-form user entries that could potentially result in errors | All forms use required fields, and when a field is not required, it is handled through defaults and error handling (which routes to an error page on the website) and through flash messages. This allows for a seamless user experience. |