**Title: Online Videogame Store**

**Smart, Darren James**

This is a Flask Web App written in Python, HTML, CSS, JavaScript, SQL and used PostgreSQL as a database managing system.

The app contains routes to a “Home” page, an “All Products” page, a “Contact” page, a product page “Product/<prod\_id>” for a product selected within the “All Products” route, a “Category” route accessed from the page of any chosen product, a “Search” route, and a “Basket”. There is also an additional ‘hidden’ route called “Easter Egg” which autoplays a videogame-themed movie.

All pages are rendered as extensions of “Layout.html” and populated with data using the Jinja 2 engine.

Custom-made functions within “image\_converter.py” were created to allow for images to be converted into bytea data and stored directly in the database. On running the app from “Online\_store\_webapp.py”, “Filefinder()” first creates a list of all image paths in /database/images/ using “os.walk()”.

“img\_to\_bytea()” with the modules “psycopg2” and “io” then connects to the database and, using a “for” loop with an “if” statement to prevent duplicates, works through the list of image paths and converts each image to bytea data before storing them in table “images” along with a name for each image extracted from the filename.

“bytea\_to\_img()” accepts image binary data and uses the “base64encode()” and “decode(‘utf-8’) “ functions from the base64 module to convert the bytea data to base64 encoded string which the browser can then interpret as an image.

“dict\_maker()” creates a dictionary of all product information, including images, for use in the “All Products” route. This is done by querying the database for all rows related to all products and creating dictionary entries where the key is the product id and the value is a dictionary of all data related to that product id; with the image binary data being converted to a usable format using “bytea\_to\_img()”. A Jinja 2 “for” loop in “all\_products.html” then unpacks all data for each dictionary product to be displayed in the page.

4 JavaScript scripts were made in this project;

“script.js” simply verifies if the browser is in light/or dark mode using “window.matchMedia” and changes the favicon to the lighter or darker version.

“image.js” was created to allow for toggling of the displayed image in the “Product/<prod\_id>” page.

“basket.js” uses an “EventListener” to update the displayed available stock after subtracting the number of items added to basket.

“total.js” uses an “EventListener” to update the displayed Total Price.

“styles.css” contains all styles used in the project.