# CS348/916 Exercise

## Objectives

A prototype for an online shopping WebApp (a website that is compatible with mobile devices) is provided in the form of some HTML, CSS and JavaScript code that allows one to retrieve information from a catalogue of products available in the shop, fill a shopping basket with items, record shipping and payment data and invoke the ordering process resulting in an invoice being generated. The main usability goal of the WebApp is that it should be equally easy to use regardless of whether it is used on a desktop computer or on a mobile phone.

The prototype WebApp has a number of usability problems. The aim of the exercise is to evaluate the prototype against a set of usability guidelines and, based on this, to redesign it to: a) eliminate the usability problems you discover; b) meet its main usability goal. The redesigned WebApp will need to be evaluated to demonstrate that a) and b) have been achieved.

## Resources Provided

An implementation of the prototype WebApp in a set of code files:

- A set of JavaScript files that contain the code used to interact with the shopping basket:
    - shop.js
    - checkout.js
    - products.js
- A set of HTML files that contain the different pages that comprise the system,
    - index.html
    - checkout.html
    - creditcard.html (there is no need to modify this file)
- A set of CSS stylesheets
    - colours.css
    - common.css
    - shopmobile.css
- Some utility files to help you get started:
    - instructions.html (giving some links to resources that can help)
    - convertColours.html (to help you modify colours.css)

The prototype system code, its description and the UI are bundled together in a compressed zip file: `eveg.zip` that can be downloaded from the module web page.

To run the application on the lab computers, we recommend using Visual Studio Code (see instructions.html for hints on how to set this up).

We recommend you use Firefox as the browser. We haven't uncovered any issues with specific browsers, but Firefox is the browser we have tested most thoroughly. Firefox is available on lab machines.

The exercise is divided into **two parts**:

1. **Initial evaluation and redesign**:
    a. Choose one usability evaluation method and apply it to the prototype WebApp.
    b. Redesign the WebApp so that it eliminates the usability problems discovered. Do not make the implementation over-complicated.

    Submit to Tabula: i) a list all the usability problems you discover, with a brief explanation for each one of why you think it is a problem; ii) a description of the method you used to identify them. Include screenshots of your revised WebApp to document the changes you have made.

2. **Evaluation of redesigned WebApp**:
    a. Conduct a user-based usability evaluation of the redesigned WebApp to test: a) it is more usable on the desktop than the prototype; b) it is as usable on a mobile phone.

    Submit to Tabula a description of the evaluation design you used and the results. Do the results confirm that your revised WebApp is more usable in both cases a) and b)?

**Deadlines**

The deadline for submitting part 1) is **week 6, Friday 16th February**.

The deadline for submitting part 2) is **week 11, Monday, 18th March.**

**Getting Started**

Lectures 4-8, chapter 14 in Rogers et al. and chapter 10 in Benyon address the challenges that you will encounter in the course of the exercise.

**Assessment**

Grading is based upon the following basic scheme:

- Up to 40% for Part 1.
- Up to 60% for Part 2.

Rob Procter, Jonny Foss, Anna Guimaraes

2023-24

# Appendix eVeg Model Interface

This document describes the API of the model underlying the eVeg shopping system.

- products.js
  - imagesArr
    - an array containing products stored in the form:

      [ProductName, Category, image path, amount, units,

      Cost in pence]
  - initProducts function
    - Converts the product array into an object that can be used by shop.js and checkout.js
    - When the products have been initialised, the callback function is called
  - setCookie
    - Sets a cookie with a name and a value
  - getCookie
    - Retrieves the value of the cookie with the specified name
- shop.js
  - cardTemplate
    - A string that can contain HTML – this string '[EVEGPRODUCT#]' will be replaced with the product ID (from the array). If there is a data-field attribute and a data-num attribute, the contents of this element will be replaced by the redraw method.
  - init
    - Sets up event listeners (functions that will be called when specific elements are clicked)
  - resetListeners
    - Sets up event listeners (functions that will be called when specific elements are clicked) for each product (when products are redrawn)
  - inputChange
    - When a product number input is changed, updates the quantity in the basket via changeQuantity
  - changeQuantity
    - Sets the value of the basket object, updates the number input, and calls refreshBasket.
  - increment/decrement
    - Calls changeQuantity to increase/decrease the quantity by 1
  - filterFunction

        - Used to filter the productDetails object, run for each product – should return true if the product needs to be shown, or false if the product needs to be hidden
    - sortFunction(a,b)
        - Helper function to determine display order (return true if item a should appear after item b)
    - redraw
        - Clears and redraws the list of products based on the card template (filtered based on filterFunction and sorted based on sortFunction). The listeners are then reset for each product so that the quantity controls for each item are activated.
    - updateQuantityInputs
        - Called after the product list has been redrawn to ensure the displayed number inputs are correct.
    - refreshBasket
        - Loops through the products in the basket and calculates the total. The basket object is then stored in the cookie, and the element with id of 'basketNumTotal' is updated with the new price. The total is also returned so that it can be used by other methods.

- checkout.js
    - init
        - initialises products and resets listeners
    - resetListeners
        - Adds a listener so that clicking on the element with the id of 'paycreditcard' directs the user to the (simulated) payment screen (such as PayPal, WorldPay, etc.)
    - calculateBasket
        - Reads the basket from the cookie and generates an invoice