

1 **DAT310 Info 2019**

The exam contains 16 questions/exercises. You can collect 100 points in total.
You may use all available online and offline resources.
But you have to solve the exam yourself, so **it is not allowed to ask others, chat, ...**
Remember, if you spend a lot of time searching for answers online, you may not have time to complete the exam.

The exam contains two parts.
Part 1 contains multiple choice and text questions.
Answers should be given using the available forms.
The scoring of multiple choice questions is as follows

- The correct solution is 2 points
- Wrong answer is -1 point

Part 2 contains several programming exercises related to example applications.
Starter files for all exercises are provided as a zip archive.
Download the starter files and use your IDE to solve the exercises.

The different exercises include forms.
If you copy past any code from online you must include a reference in the form.

To submit your solutions to part 2, submit a single zip file, containing the starter files and your added code.
This zip file can be uploaded on the last page of the exam.
Code added in the individual solution forms will not be considered if a zip file is available.
This should only be a last resort.

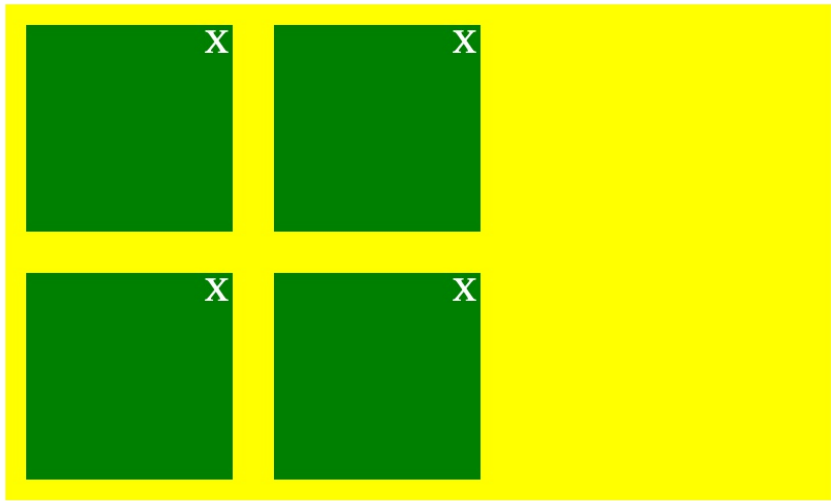
If you have comments to any of the exercises, you may write them here

Maximum marks: 0

12

CSS positioning

QUESTION



Add some css rules to the file **css_positioning/index.html**. *(4 points)*
Make the page look like the image above.
You can find the file in the zip arkive in the map css_positioning.
Do not change the HTML code.
Upload the file including the added rules as part of the zip arkiv on exercise 18.

Make sure you include the following:

- the green divs lie in 2x2
- the x's are white and lie in the upper right corner of the green div's

List source here

Maximum marks: 4

13

jQuery coding

Implement additional functionality for the jQuery timer. *(10 points)*
In this exercise you have to implement additional functionality for a jQuery timer.
You can find the timer application in the zip arkive in folder **timer/**.

Add the following functionality:

Buttons: *(see images 1 and 2 below)*

- Add a *cancel* button to the timer. When clicked, the timer disappears and the countdown form appears, allowing to specify a new timer.
- Add a *stop* and *continue* buttons to the timer. When clicked the *stop* button halts the countdown. The *continue* button allows to continue the countdown.
- When the countdown is running, the *continue* button should be hidden.
- When the countdown is stopped, the *continue* button is visible but the *stop* button is hidden.

Countdown title *(see images 3, 4 and 5 below)*

- Add a form field to the countdown form that allows to specify a title for the countdown.
- Convert the title to only capital letters (upper case) and display it as part of the countdown.
- When the countdown reaches 00:00 display the title to the user in an alert.

DAT310 examen 2019

You can but you do not have to use jQuery.

List kilder her

Image 1: *stop* and *cancel* buttons

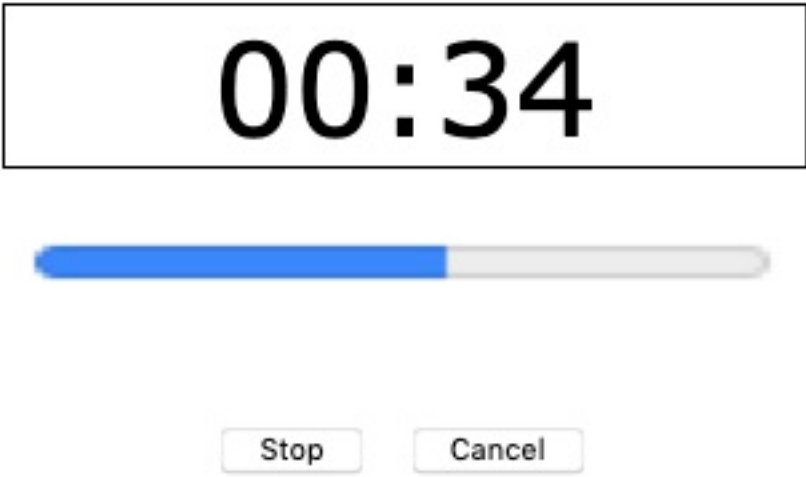


Image 2: *continue* and *cancel* buttons

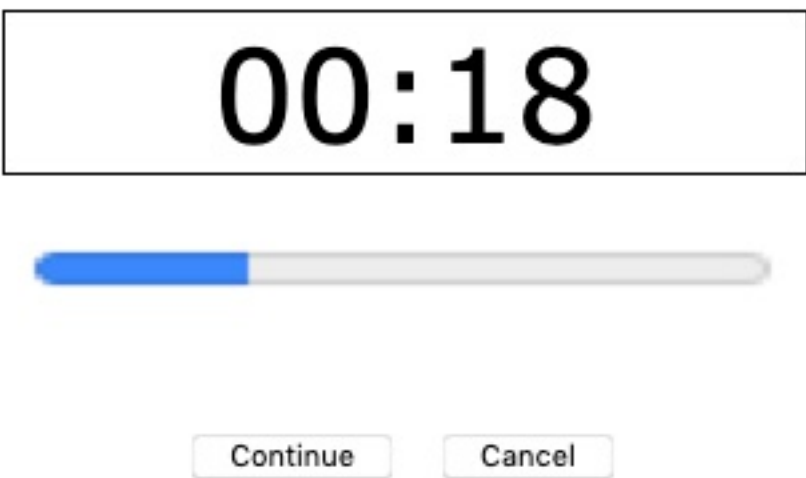


Image 3: *Countdown form* with title form field

1 minute

⬆⬇⬆

Go!

Image 4: *Countdown title* displayed



Image 5: Alert



Maximum marks: 10

14 JS coding - todomlist

Implement additional functionality for a todo-notes application. (24 points)

In this exercise you have to implement additional functionality for a JS application for creating a todo-list. The current application allows the user to add new entries with title, date and content to a TODO list.

You can find the todo-list application in the zip arkive in folder **todolist/**.
You have to add code to both index.html, todos.js and possibly a new CSS rule.

Implement the following additional functionality:

Fomat display:

- Implement the function *formatDate* in the **todos.js** file. The function ensures that the date for new added todos is formatted dd/mm/yyyy (e.g. 17/05/2019).
 - Hint: *It is enough to use string formating and not necessary to convert to a datetime object.*

Add Categories: (See Image 1 and 2 below for an example)

- Add a dropdown select to the AddTodo form. The select should have a label *Category* and allow to select one out of the categories *home*, *work*, *other*.
- When adding a new entry, the category should be stored in the *category* of the new object.
- The category should be displayed in the upper right corner of the todo div element.
- Todo entries with category *home* should have a yellow background and border.
- Todo entreis with category *work* should have a blue background and border.

Add data:

- Update the **data.js** file. Add two more entries to the *todos* array with categories *work* and *other*.

Show/hide categories: (See Image 3 below.)

- Add a dropdown select to the *control* div with categories *all*, *home*, *work*, and *other*.
- When *home*, *work*, or *other* is selected, only todo entries with this category should be shown.
- When *all* is selected, all todo entries should be shown.

Parse multiple todo point from the input: (See Image 4 and 5 below.)

- Add the following instruction to the *Todo* textarea in the addentry form:
 - "Separate todos by semicolon: todo1; todo2; ..."
- The instruction should show when the textarea is empty.
- Update the function *Entry()* constructor in **todos.js** to split the *content* by semicolon, resulting in an array of todo points.
- Remove trailing white space from the strings in *content*.
- Update the function *displayEntry*(idx) in **todos.js** to display the different elements from the *content* array as an unordered list.
- Make sure no empty list elements are displayed.

List sources here

Image 1: category select dropdown

Add Todo

Title:

Category:

Date:

Todo:

Home

✓ Work

Other

Add

Image 2: colored todo elements

Add Todo

Title:

Category:

Date:

Todo:

Other

mm / dd / yyyy

Separate todos by semicolon: todo1; todo2; ...

Add

Cleaning

16/05/2019

#home

• clean living room

• clean kitchen

A work todo

29/05/2019

#work

Another todo

30/05/2019

#other

Image 3: Select to show/hide categories

Display Category

✓ All

Home

Work

Other

Add Todo

Title:

Category:

Date:

Todo:

Other

mm / dd / yyyy

Separate todos by semicolon: todo1; todo2; ...

Add

Add Todo

Title:

Todo with bullet list

Category:

Other

Date:

05 / 30 / 2019

Todo:

Point 1; point2;
point3;

Add

Image 5: The Todo with bullet points

Todo with bullet list

30/05/2019

#other

- Point 1
- point2
- point3

Maximum marks: 24

15

Phoneshop main page

In this, and the following exercises you need to complete a online shop (flask) application. Starter files for these exercises are available in the zip archive in the folder **/phoneshop/**

Bootstrap formating (4 points)

A static version of the stores main page is available as */phoneshop/static/index_static.html*. The file includes the bootstrap css library.

Add classes to the file to make the design responsive, achieving the following:

- On large screens, 3 articles should be dispayed per row.
- On medium screens, 2 articles should be displayed per row.
- On mobile phones and very small screens, only one article should be displayed per row.
- The jumbotron should be hidden on mobile devices and very small screens.

See the images below.

Jinja templating (4 points)

Compete the template at *templates/index.html* that is displayed as main page of the flask application. The resulting page should look the same as the static page on *static/index_static.html* with the changes introduces above. Also:


- The template *index.html* should extend *layout.html*
- Use a for loop to display the articles. The data for the articles is available in the flask application as *phones* which is imported from *phones.py*.

List sources here

Image 1: Large screen

have fun
with a **paper phone**
all you need to build your phone is some scissors and glue


Products



LG G7
A great paper-phone with interesting branding.

210,-


Details



Samsung Galaxy S10
A paper-phone with some tricks up its sleeve.

200,-


Details



iPhone X
must have!

250,-


Details



Google Pixel 3
Love it or hate it.

210,-


Details



Google Pixel 2
Get an old school paper phone!

190,-

Details



LG G6
a back-to-basics design that focused on being a great paper phone!


195,-

Details

Image 2: Small screen

have fun
with a **paper phone**
all you need to build your phone is some scissors and glue


Products



LG G7
A great paper-phone with interesting branding.

210,-


Details



Samsung Galaxy S10
A paper-phone with some tricks up its sleeve.

200,-


Details



iPhone X
must have!

250,-


Details



Google Pixel 3
Love it or hate it.

210,-


Details



Google Pixel 2
Get an old school paper phone!

190,-

Details



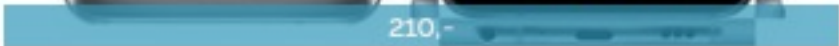
LG G6
a back-to-basics design that focused on being a great paper phone!

195,-

Details

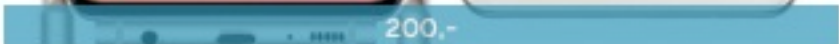
Image 3: Mobile screen

Products



LG G7

Details



Samsung Galaxy S10

Details

Maximum marks: 8

16 **Phoneshop details**

In this exercises you need to complete the details page of the flask application.

Starter files for these exercises are available in the zip archive in the folder **/phoneshop/**

Currently any of the *details* links on the main page lead to the same details page (for LG G7).

Update the /details route and links (4 points)

- Introduce a parameter to the */details* route that is used to display details for different phones.
- Update the *details()* function to take a parameter, find the corret phone in the *phones* array and render the details.
- Update the links in your *index.html* template to link to the corret details page.

*Hint: You could use the **id** entry of the phone in the phones array as parameter.*

You do not need to change the *details.html* template.

List sources here

Maximum marks: 4

17 **Phoneshop cart**

In this exercises you need to complete the shopping cart page of the flask application.

Starter files for these exercises are available in the zip archive in the folder **phoneshop/**

First some info on the existing code:

There are three ways to access the */cart* route.

1. Via the *Shoppingcart* link in the nav bar, using a GET request.
2. Via the form on the */details* page. (POST)
3. Via the form on the */cart* page. (POST)

Case 2 and 3 are differentiated using the hidden form field *action*.

Hint: If you have trouble with the session, you can work on the buttons and message flashing without the session functionality.

Storing the cart in the session (6 points)

Update the cart() function:

- Store the content of the cart in the session as a dictionary containing (name, quantity).
- Retrieve the cart from the session to display it.
- Delete or empty the shopping cart on checkout (*do_3*).

Buttons to change quantity *See image 1 below. (15 points)*

Implement the following functionality for the +/- buttons on the cart table.

You can use the js file */static/script.js* to implement JS functionality.

- Load the */static/script.js* fil on the *cart.html* template.
- Use the *url_for* function to load the script.
- The buttons should be hidden after confirmation (*do_2* and *do_3*)
- When the + button is clicked, use JS to increase the count shown in the table.
 - Hint: *You can use the value in the for attribute of the button to find the id of the count.*
- When the - button is clicked, use JS to decrease the count shown in the table if it is larger than 0.
- When changing the count displayed in the table, also send an AJAX call to the server to change the cound or quantity stored in the session.

Flashing messages See Images 2 & 3 below. (6 points)

The `cart()` function in `app.py` contains 3 messages that may be flashed.

- Extend the `cart.html` template to display flashed messages.
- Use bootstrap `alarm` classes to style the messages, success message green, error message red.
- The message should contain a button/link X that can be used to close the message.

Cart pill: See Image 4 below. (3 points)

Add a bootstrap pill to the `Shoppingcart` link in the `layout.html`.

- The pill is displayed on the main (`Store`) and details page.
- The pill is only displayed if the shopping cart contains some items.
- The pill shows the number of items in the shopping cart (as stored in the session).

List sources here

Image 1: Confirmation (`do_2`) without +/- buttons

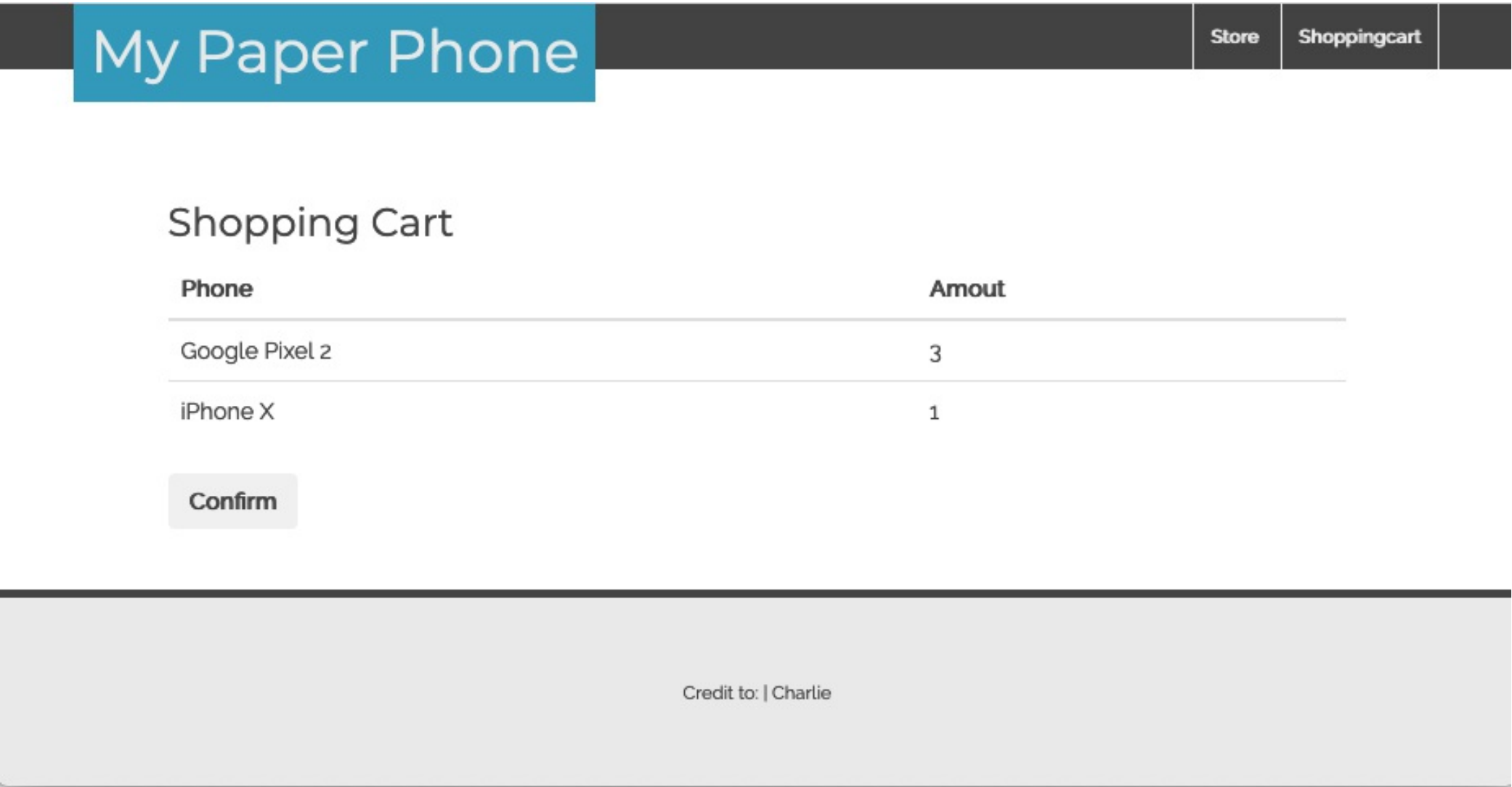


Image 2: Thanks (`do_3`) without +/-, with success message

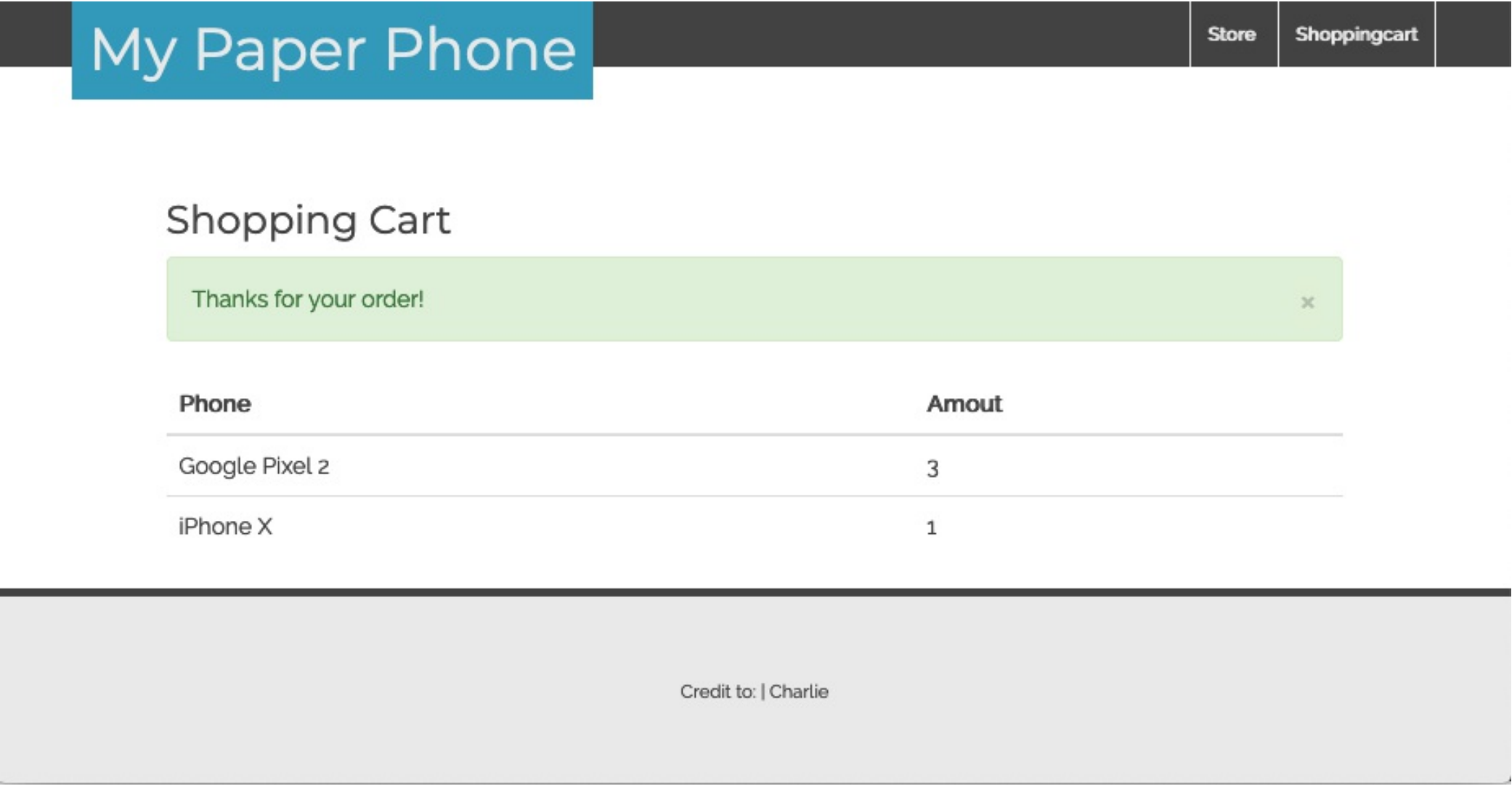


Image 3: Cart with error message:

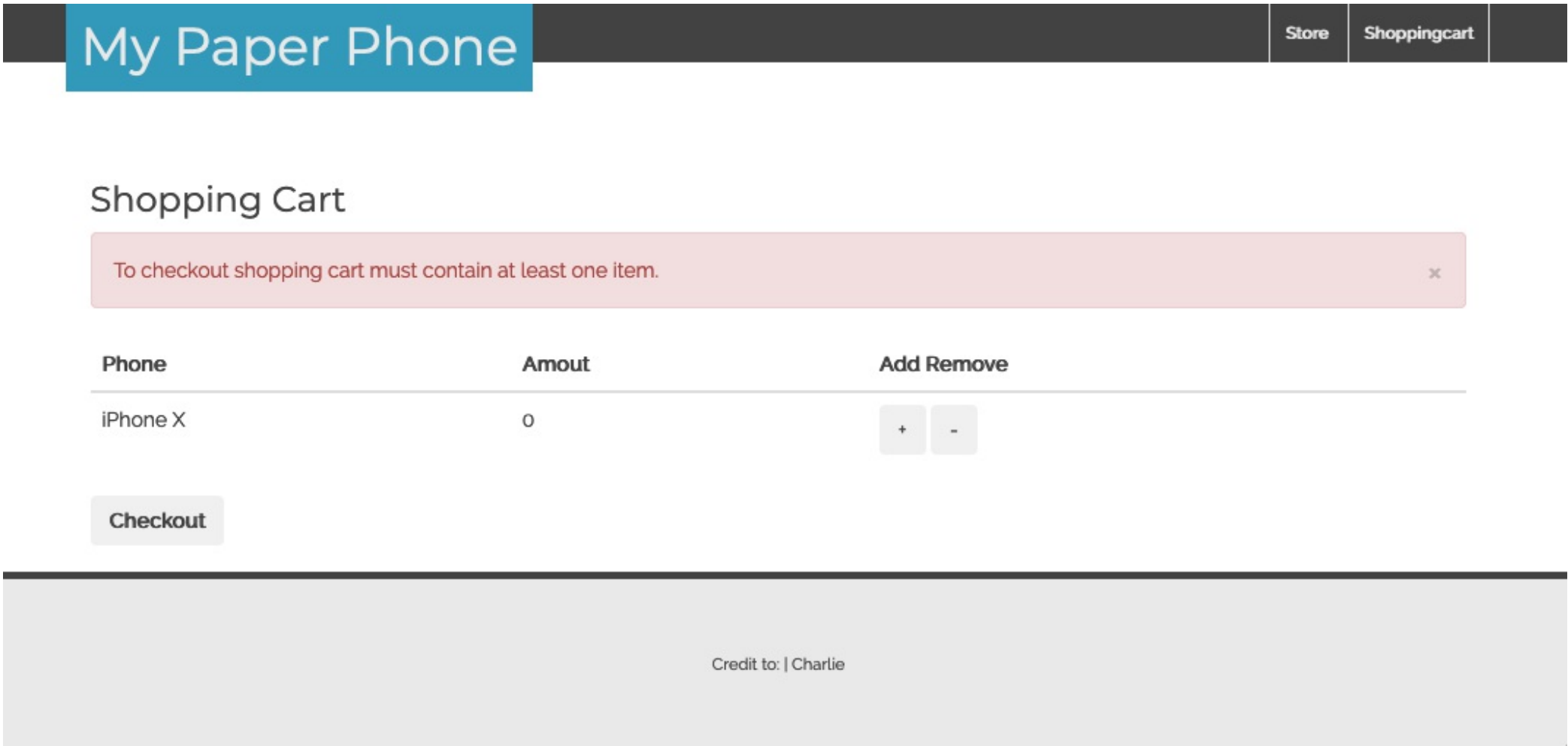


Image 4: Shoppingcart link with pill for 3 items



Maximum marks: 30

18 Upload

Create a zip archive from the hole *files* folder and upload it here:

Maximum marks: 0