Task 1:

Registration Scenario (Success):

Working files: registration.page.feature, registration.page.spec.js, registration.page.js

Linking files: main.page.feature, main.page.spec.js, main.page.js, login.page.js, helper.js, locator.js, common.spec.js, page.map.js

- Create a user account with random generated data and use the registration function to register an account for this user

  Task plan:

  - Follow the registration page from the main page via login page.
  - Input randomly generated data to input the required data during registration
  - Successfully register user.

  Implementation:

  - Following the main.page.feature implementation I started the journey from the main page.
  - Then by clicking to the “Register button” through login page I reached the “Registration page”.
  - I wanted to implement the “randomly generated data” and feed those as input but I guess I need more time to understand how to put the data while registering a user and also preserve the data to use further during login.
  - Register successfully

  Improvement:

  - Randomly generated data feeding as input. I plan to use a database and put generated data there so that the data can be used in case of any scenario
  - Checking whether “Registration is successful” needs to be more convincing. I am not happy with the way I handled it. Profile checking and verifying the provided data from the profile would be more convincing.

Task 2:

Login Scenario (Success and Failure):

Working files: login.page.feature, login.page.spec.js, login.page.js

Linking files: main.page.feature, main.page.spec.js, main.page.js, helper.js, locator.js

Task Plan:

- From main page go to login page using “Register button”
- Provide information(valid/invalid)
- Check login status

Implementation:

- Entered login page by clicking on the “Register button”
- Enter login information. Used table to feed the login data in the input fields.
- By using table I could check both failed and successful login scenario

Improvement:

- The code could be more dynamic
- Randomly generated login data could be used if they were implemented during registration

Task 3:

Working files: shopping.page.feature, shopping.page.spec.js, shopping.page.js

Linking files: main.page.feature, main.page.spec.js, main.page.js, login.page.js, helper.js, locator.js, wishlist.page.js, search.page.js, search.page.spec.js, common.spec.js, page.map.js, basket.page.js

1x Shopping Scenario:

- Go to the website and select 5 items from the Webpage which will be added to your Wishlist
- Go to your Wishlist and add all 5 existing items of the Wishlist to your basket

Task Plan:

- Login as valid user
- Search/browse some items
- Add 5 items in the wishlist
- Go to the wishlist page
- Input zip code(both correct and incorrect)
- Add the added items to the shopping cart

Implementation:

- Login as valid user from the login page
- Input “bed” in the search field and search items
- Added 5 items to the wishlist
- Went to Wishlist page by clicking wishlist button
- Entered a valid zip code
- Clicked “Add all item to the basket”
- Checked that user successfully arrived in the basket page

Improvement:

- Add items in the wishlist by browsing different categories

    - The Gherkin code for item addition in the wishlist can be more dynamic.
    - Need verification of the zip code as well as the scenario without giving a zip code.
    - Need scenario of adding item from wishlist to basket on by one.

P.S: I used the old project file provided to my during live code session. I tried to use the new file but apparently, cypress 10 has stopped working in my pc for some strange reason as my older projects are also not working.

I tried to understand the framework as much as possible within the given time frame. I think some more time could have given me a better understating of the framework and apply all the aforementioned improvements.