In step 5, you are asked to calculate the cross-entropy loss of the model and use it to update the model. But it is the gradient of this loss with respect to the output layer's activations that is directly used in backpropagation, not the loss value itself.

Also, we give a flow of Step 4-7:
Firstly, let's clarify each part of the network according to the requirements:
$Input\ x \Rightarrow z_1 = W_1 x + b_1 \Rightarrow a_1 = sigmoid(z_1)$
$\Rightarrow z_2 = W_2 a_1 + b_2 \Rightarrow a_2 = sigmoid(z_2)$
$\Rightarrow z_3 = W_3 a_2 + b_3 \Rightarrow output = softmax(z_3)$
Note that z_i, a_i are all the units in the neural network; W_i, b_i are the parameters of the ith layer's linear function.
Then here're the gradients of each layer You may refer to the slides 36-38:

1 Gradients of Weights and Biases in Output Layer (Layer 3):

- Gradient of the loss L with respect to weights W_3: $\frac{\partial L}{\partial W_3} = \frac{\partial L}{\partial z_3} \cdot a_2^T$
- Gradient of the loss L with respect to biases b_3: $\frac{\partial L}{\partial b_3} = \frac{\partial L}{\partial z_3}$, where $\frac{\partial L}{\partial z_3} = \text{softmax}\,(z_3) - y$.

2 Gradients in Hidden Layer 2:

- Gradient of L with respect to W_2: $\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial z_2} \cdot a_1^T$
- Gradient of L with respect to b_2: $\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial z_2}$ Where $\frac{\partial L}{\partial z_2} = \left(W_3^T \cdot \frac{\partial L}{\partial z_3}\right) \odot \text{sigmoid}'\,(z_2)$.

3 Gradients in Hidden Layer 1:

- Gradient of L with respect to W_1: $\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial z_1} \cdot x^T$
- Gradient of L with respect to b_1: $\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial z_1}$ Where $\frac{\partial L}{\partial z_1} = \left(W_2^T \cdot \frac{\partial L}{\partial z_2}\right) \odot \text{sigmoid}'\,(z_1)$.
  $\odot$ denotes element-wise multiplication

Initialize learning_rate, epochs, and batch_size according to the requirements.
Calculate number of batches (n_batches) based on the size of training dataset.
Initialize an empty list for tracking losses.

For each epoch (total of 'epochs' given above):
- Shuffle the indices of the training images
- For each batch in the total number of batches:
  - Select batch data (images and labels) using shuffled indices
  - Feedforward:
    - Compute results for each layer by the previous structure:

(wx+b => sigmoid) => (wx+b => sigmoid)=>(wx+b => softmax)=>results

- Backpropagation:
  - Compute error at output layer. Note the last hint on how to calculate it.
  - Compute gradients for weights and bias in each layer, **note that we use mini-batch gradient descent here. So, you may need to modify the above gradients a little.**
  - Update weights and biases using the computed gradients and learning rate
- At the end of each epoch:
  - Compute forward pass on the entire training dataset to calculate loss
  - Log the epoch number and the loss
  - Evaluate the model on the test dataset:

    Perform a forward pass using test data

    Compute accuracy by comparing predicted labels with actual test labels

    Print accuracy for the test dataset to help you monitor the training process

    (Note that in the real application you **cannot use testing dataset** to monitor the training process. You will need to split a validation set from the training test to finish this step. But we don't ask it in the homework for simplicity)