



Technical Writing Samples

February 20, 2024

Rebecca Ellis

1

Sample 1:

1.1 API reference document to explain to a Java developer how to call the needle in a haystack method:

```
public static void findNeedles(String haystack, String[] needles) {
```

```

if (needles.length > 5) {
    System.err.println("Too many words!");
} else {
    int[] countArray = new int[needles.length];
    for (int i = 0; i < needles.length; i++) {
        String[] words = haystack.split("[ \\\"'\t\n\b\fr]", 0);
        for (int j = 0; j < words.length; j++) {
            if (words[j].compareTo(needles[i]) == 0) {
                countArray[i]++;
            }
        }
    }
    for (int j = 0; j < needles.length; j++) {
        System.out.println(needles[j] + ": " + countArray[j]);
    }
}
}

```

haystack: [string] contains a string of unknown length needles: [list of strings] contains a list of strings

countArray: [list of integers] used to keep track/count the number of times each needle item occurs

words: [list of strings] contains the words that are in haystack

//Function: findNeedles(string)(list of string) -> string:integer

// Purpose: The purpose of this function is to evaluate the total number of occurrences of each item in the list, needles, that occur in the string, haystack.

The purpose of the function findNeedles is to find a string inside another string. It can be applied, for instance, as in the above script, to find where a certain word is within a sentence.

The function findNeedles accepts two parameters, a string (haystack) and a list of string values (needles).

2

First, an if-else condition is established within the function. The if statement checks the number of elements in the list of strings, needles, beginning with the case CountArray, by using the needles.length method. If the list of strings, needles, contains more than 5 items, the error “too many words!” is output to the console. So if the algorithm detects more than five words within a sentence, it returns the error.

1.2 Assume you have a chance to send comments or questions to the person who wrote the

code. Suggest ways to improve the code, for example, to reduce memory usage or enhance features.

1. To define the arguments, start with wrapping, creating the main method and calling your own method thereafter with the suitable arguments.:
2. Why limit the needle lists to 5 strings? Save memory and use a counter variable so that the array of text (haystack) is created only once and the split function is called before the for loops, as shown below, instead of with each new word search:

```
public static void findNeedles(String haystack, String[] needles) {
    int[] counter = new int[needles.length];
    String[] words = haystack.toLowerCase().split("[ .,:;\"'\\t\\n\\b\\f\\r]",
    0);
    for (int i = 0; i < needles.length; i++) {
        for (int j = 0; j < words.length; j++) {
            if (needles[i].equals(words[j])) {
                counter[i]++;
            }
        }
    }
    for (int j = 0; j < needles.length; j++) {
        System.out.println(needles[j] + ": " + counter[j]);
    }
}
```

3

See [Alameda Tech Lab](#) for more information.

3. Consider scalability: what if some or all strings are very large? Use Java's regex utility and Python generator and include a while statement in the function:

```
public static int countOccurrences(string haystack, string regexToFind) {
    Pattern p = Pattern.compile(regexToFind);
    Matcher m = p.matcher(haystack); // get a matcher object
    int count = 0;
    while(m.find()) {
        count++;
    }
    return count;
}
```

See [Stack Overflow](#) for more information.

This should use far less memory than .split for a relatively large 'haystack'.

4. Create statement for the haystack function to address null pointer exceptions, meaning what happens if there is no search term (no needle) or nothing to search it in (no haystack), something like:

```
if (needle == null || haystack == null || haystack.isEmpty())  
    return false;  
if (needle.isEmpty())  
    return true;
```

1. How much of the content did you write? I wrote 65% of the content (I am not the author of the code).
2. Does the document represent your original writing, or is it existing content that you revised? Researched and revised writing.
3. Was the document edited by other people - No
4. Where did you get the information to write the document? Swagger, Stackoverflow, Medium, Google search
5. Share how you obtained any code samples. See above, sandbox testing.
6. Was a company style guide used to write this document? Yes, the first prompt question.
7. Provide any additional useful context for the sample, such as deadlines, achievements, etc. N/A
8. Was this document part of a larger documentation set? › No.

4

Sample 2:

Tic-Tac -Toe is a simple game that even children can play and you can play it anywhere.

All you need is:

- A flat surface, such as piece of paper, a wall, or sandy ground, and
- An instrument for drawing, such as a pencil, pen, marker or a stick.

To play the game, you need a partner to play against. The game involves a strategy of looking ahead and trying to figure out what your partner might do next.

RULES FOR TIC-TAC-TOE


1. The game is played on a grid that's 3 squares by 3 squares (9 squares total). Each round of the game starts with 9 *empty* squares.

2. To start, draw the grid on the paper or flat surface.
3. Decide with your partner who will mark Xs and who will mark Os in any square on the grid, one square, or **turn**, at a time. Once a player has chosen their X or O, each player keeps the same X or O for all their turns until there is a **win** or **tie**, marking the end of a **round** in the game.
4. The X player begins the game, taking the first turn in a round. The O player takes the second turn in a round. Players alternate their X and O turns by putting their respective marks in any of the empty squares, one square at a time for each turn.
6. The game continues with each player taking their turn to fill one empty square until:
 - All 9 squares in the grid are filled with Xs and Os. In this case, no one wins the game and this is called a **tie**.

Or

 - One row, column or diagonal of three squares in the grid is filled with three of the same mark, that is, it will be all Xs or all Os. In this case, the player who filled the row, column or diagonal in the grid (up, down, across, or diagonally with each turn) with the same X or O **wins** (meaning, the opposing player who did not manage to get an entire row of the same X or O loses the round).. It is customary for the winner to draw a line through the row, column or diagonal to underscore their win.

5

- 
7. In either a **win** or a **tie**, the players can opt to start a new round of the game with a fresh, empty 3x3 (9-square) grid, playing as many times as they want.

TIC-TAC-TOE STRATEGY

Part of your strategy is trying to figure out how to get three Xs or three Os in a row, column, or diagonal. The other part is trying to figure out how to stop your partner from getting three Xs or Os in a row.

After you put an X or an O in a square, you start looking ahead. Where's the best place for your next X or O? You look at the empty squares and decide which ones are good choices — which ones might let you make three Xs or three Os in a row.

You also have to watch where your partner puts their X or O. That could change what you do next. For example, if you have chosen X for the round, your partner puts two Os in a row, you

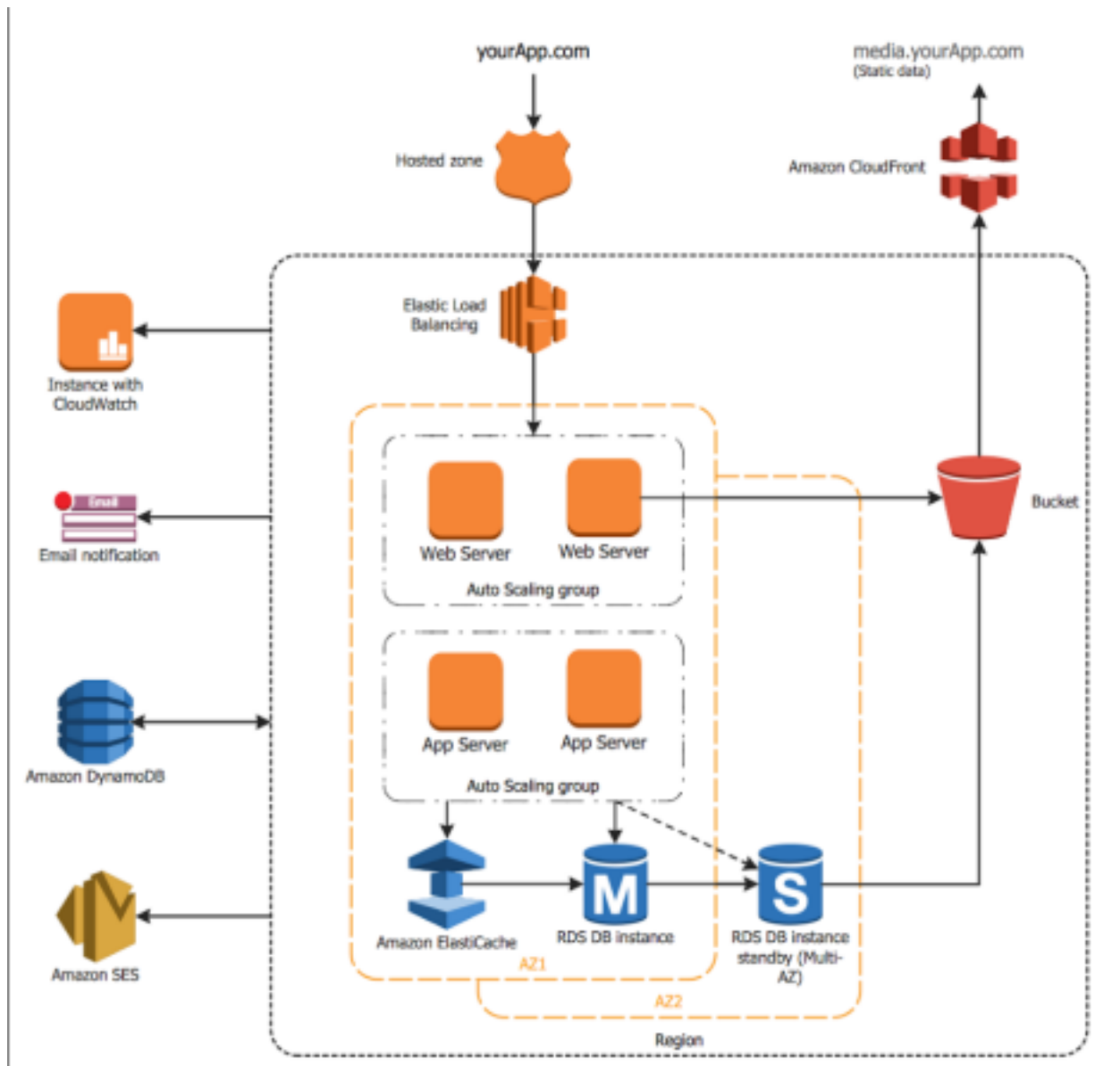
have to put your next X in the last empty square in that row, or your partner will likely win.

If you always pay attention and look ahead, you are more likely to win or at least tie a game of Tic-Tac-Toe. If you are not paying attention and you do not think ahead, you are more likely to lose in Tic-Tac-Toe.



Sample 3:

3.1 Below is an architecture diagram of a fictional product. Based on this diagram, I wrote an overview of this product.



7

YourApp app uses several services to manage developer workflows in Amazon Workspace (AWS). One service the app accesses is to store and retrieve data from the managed NoSQL database service DynamoDB. Another service the app uses is the scalable Amazon Simple Email Service (SES), so that developers can send emails from the application to handle transactional, marketing, or mass email communications. The app also outputs to an Amazon Instance with CloudWatch, which is used to monitor the app data and provide usable insights on the app's resource usage in the form of logs, metrics, and events, providing a dashboard to allow developers to visualize this data.

YourApp resides in a hosted zone at the URL YourApp.com (the website), which can

be accessed anywhere in the world. It is a dynamic application developed in Amazon Workspace (AWS) that handles redundancy and heavy load via front-end facing global Elastic Load Balancing to ensure that the app is highly available to users. The application backend is located in two zones in a region. The four web servers, Amazon ElastiCache and an RDS DB Instance are hosted in one Amazon zone, and an RDS DB Instance Standby is located in the other.

There are several ways the user data is output as static media to the user.

- Under conditions of normal or low usage, the user data uploaded on the website, then sent to the backend data storage bucket.
- Under conditions of heavy usage, the four web servers reside in two autoscaling groups of two web servers each to handle dynamic load, where the data is either cached in the Elastic Cache or output to the RDS DB instance.
- There is even a multi-zone RDS DBS Instance Standby backup in the second zone as a failover for extreme heavy usage. This ensures high availability over the entire region and quick response time.



3.2 Describing API resources.

3.2.1 The following three tables show examples of requests and responses of an API resource. Using these snippets, describe this resource.

- I briefly described the request GET `http://users.api.com/USER_ID`.
- I describe in detail the response object in detail.

Request	GET <code>http://users.api.com/67923</code>
Response	<pre>{ "id": 67923, "username": "aura12", "firstName": "Aura", "lastName": "Burk", "userStatus": "deleted" }</pre>

Request	GET <code>http://users.api.com/1234567</code>
---------	---

Response	<pre>{ "id": 1234567, "username": "alex_prince", "email": "a.prince@gmail.com", "firstName": "Alex", "lastName": "Prince", "userStatus": "active" }</pre>
----------	---

Request	GET http://users.api.com/986428051
Response	<pre>{ "id": 986428051, "username": "p_king", "birthday": "01/01/1985", "firstName": "Patrick", "userStatus": "active" }</pre>

9

The resource is a user database, used to store and transmit data objects related to the users. The data objects are stored in a sparsely populated table of user data, using a GET request to query each user's data that is available in the database. The response object is the data available returned in JSON.

3.2.2 The following table shows another example of request and response of the resource http://users.api.com/USER_ID. Complete your description of the response object.

Request	GET http://users.api.com/72940168
---------	---

Response

```
{
  "id": 72940168,
  "username": "miros",
  "birthday": "24/12/1980",
  "email": "sara_14@yahoo.com",
  "firstName": "Sara",
  "userStatus": "active",
  "likes": [
    {
      "id": 154323,
      "name": "Game of Thrones",
      "about": "Watch full episodes on HBO",
      "can_post": false,
      "fan_count": 18088271,
      "genre": "drama",
      "is_verified": true,
      "network": "HBO",
      "season": "5",
      "website": "http://www.gameofthrones.com/"
    },
    {
      "id": 963297,
      "name": "The Hateful Eight",
      "about": "The official page for THE HATEFUL EIGHT",
      "can_post": false,
      "directed_by": "Quentin Tarantino",
      "fan_count": 331494,
      "release_date": "25/12/2015",
      "studio": "The Weinstein Company",
      "website": "http://www.thehatefuleight.com/"
    },
    {
      "id": 56027,
      "name": "Sola Cafe",
      "about": "We make everything with the freshest ingredients.",
      "can_checkin": true,
      "can_post": true,
      "checkins": 1430,
      "fan_count": 2130,

```

10

```
    "food_styles": ["Breakfast", "Italian", "Vegan", "Vegetarian"],
    "location": "290 W Kagy, MT",
    "place_type": "restaurant",
    "price_range": "$$"
  },
  {
    "id": 359045,
    "name": "Peter Fox",
    "can_post": false,

```

```
    "current_location": "Berlin",  
    "fan_count": 765398,  
    "genre": "hiphop",  
    "record_label": "Warner Music Group Germany",  
    "website": "www.peterfox.de"  
  }  
]  
}
```

The resource of this API is a backend user database for a web application located at the URL `users.api.com`. In this case illustrated above, the API is using the GET method to retrieve user information from the application's backend user database over the endpoint `72940168`, which is the unique key identifier `"id"` for the user record whose `"username"` is `"miros"`. The information returned in the JSON response object contains information of other profiles this specific user `"miros"` has interacted with by submitting `"like"` to different profiles in the web application, ranging from a business site or a cafe to a music artist.

Each profile record is returned in the JSON object as a list value in the `"likes"` name of the `"miros"` user record. Each profile record likewise has a unique key `"id"` identifier in the web application's backend database. However, each profile record contains name-value pairs notably distinct from those in the user record retrieved via the endpoint illustrated here `72940168`.

This is apparent in the JSON response. For instance, each profile record listed under `miros'` likes contains `"name"` instead of `"username"`. The profile records listed also contain information such as `"fan_count"` as well as other parameters determining how application users like `"miros"` can interact with them, such as ability to post content or check in.

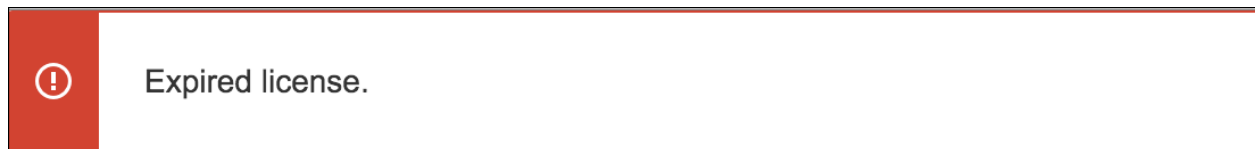
The API thus provides the interface between a user record in the `users.api.com` resource and other profile information which may be located in another resource, associated with each user as data under that user's likes. A user could theoretically have a profile as well, in which case both records would be associated with the same unique `"id"` identifier. Depending on how the web application is designed, profile

11

data could be stored in a different resource, perhaps located at a URL `profiles.api.com`, in which case the API could access it at the same endpoint (`PROFILE_ID`). So in the case that the user also has a profile, the endpoint at both resources `profiles.api.com/PROFILE_ID` and `users.api.com/USER_ID` would be the same value, thus linking the tables `USERS` and `PROFILES` in the application's backend database via the same unique `"id"` identifier.

3.2 Revision and rewriting of a user interface presented by a fictional development team. I reviewed them and rewrote them, if needed, providing justifications for my revisions of the messages, or otherwise, why I left them as-is.

3.2.1

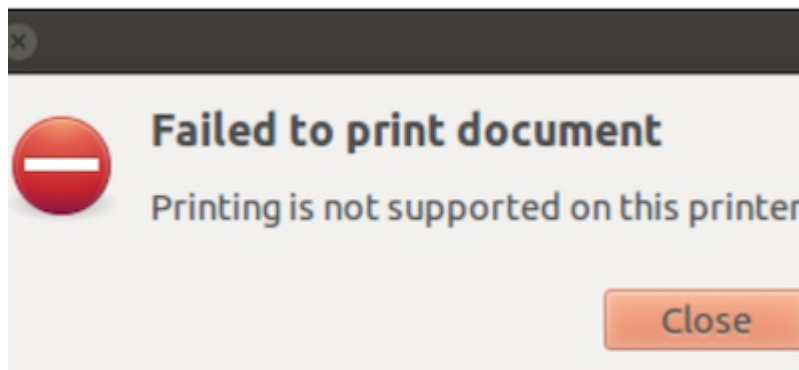


Revised message: Remove period/ full-stop.

Justification: Not a sentence.

Suggestion for improvement: Add a call to action on how to renew the license.

3.2.2



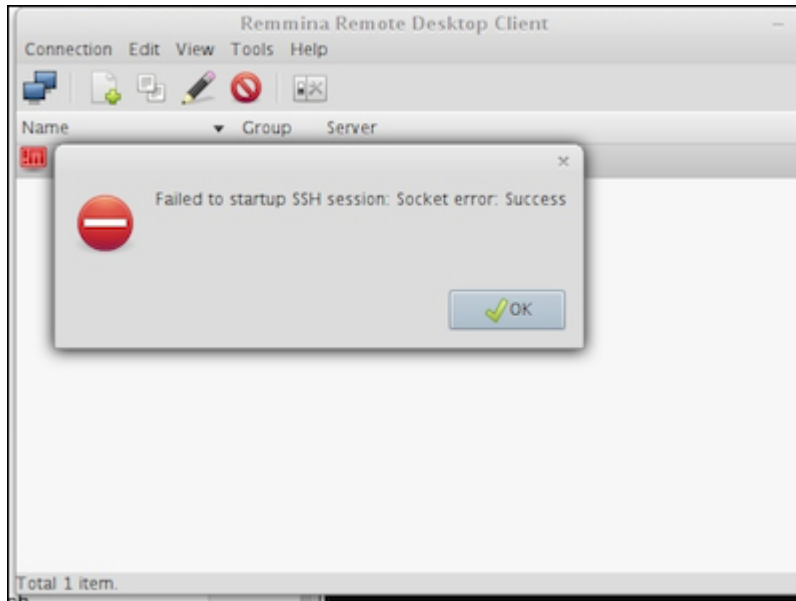
Revised message: Failed to print document.

Justification: Punctuation missing.

Suggestions for improvement: The second part is not needed, it is obvious from the above statement. How would it know which printer anyway? More concrete reasons

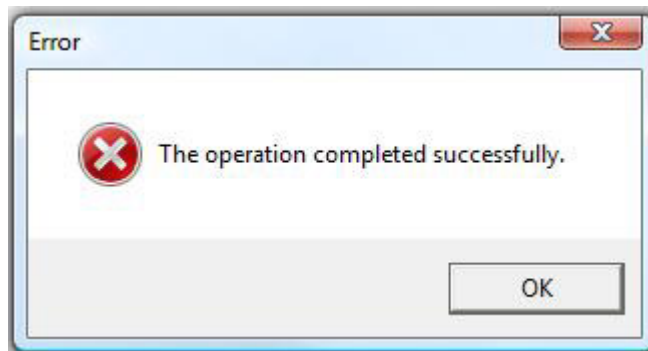
and suggestions for action should be given instead, such as “No communication with printer” or “Add printer.”

3.2.3



Suggestions for improvement: This dialog sends a confusing message. How can a failed operation be a success? Button should say Close instead of OK, as OK with the green checkmark implies that it worked.

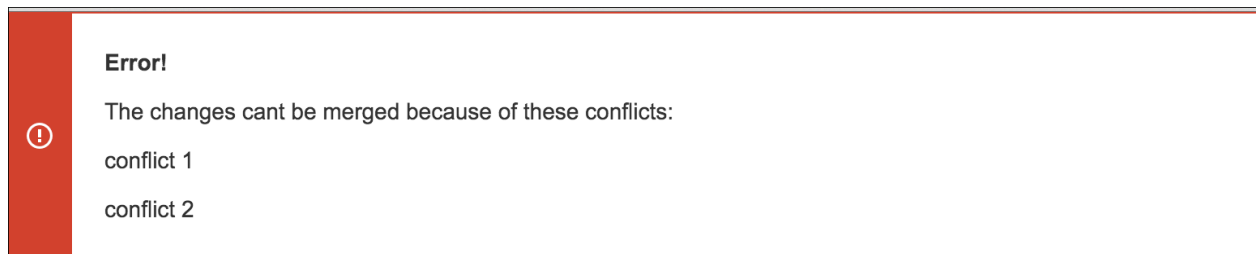
3.2.4



Revised message and suggestion for improvement: Instead of a red X it should be a green checkmark next to the OK on the button itself.

Justification: Analogous to the above, this sends a mixed message.

3.2.5



Revised message and suggestions for improvement: The changes cannot be merged because of these conflicts.: display the string values behind conflict 1 and conflict 2, not the key values. Where is the Close button?

Justification:

Confusing and missing punctuation.