1. Variables

In our code, we only declare variables close to where they are first used, and each variable is initialized when it is declared. We keep the variables' and HTML tags' spans as short as possible. Our code uses each variable for only one purpose to avoid confusion and to maintain shorter spans. We use variable names that are searchable and are within the 8-20 character range.

```
$season = $_SESSION["season"];
$budget = $_SESSION["budget"];
$travelAct = $_SESSION["travelAct"];
$q = "SELECT name, details, country, activity1, activity2, activity3, activity4, image, link,
likes, dislikes FROM Locations WHERE season = '$season' AND budget = '$budget' AND
travel_activity = '$travelAct';";
$result = $db->query($q);
$row = $result->fetch_assoc();
```

In the code above, the spans of the variables q and result are both 1. The longest variable name is 9 characters.

2. SOLID Foundations

Our code avoids duplication as much as possible. In our CSS code, we use the same styling sections for more than one element of HTML rather than creating a styling section for each element.

```
a:link, a:visited {
    color: black;
    text-decoration: none;
}
```

Single Responsibility Principle: While we do not have classes because we coded in HTML, PHP, and CSS, we still used the single responsibility principle in other ways. Each HTML/PHP page has only one reason to exist and primary responsibility. No page has the responsibility of displaying more than one thing – the homepage is only the homepage, each preference question has a different page, and the destination page has its own page.

Open/Closed Principle: Other than updating the interface, adding new functions should not require modifications and should not break elements within the other code. Nothing within the other code should break because each element is separate from the other elements wherever possible, which also allows our code to be more orthogonal.

3. Flexible Code

Our code uses concurrency for the destination recommendation page; rather than retrieving the location details one at a time where it is needed in the HTML code, the details are retrieved all at once in the php code as shown in the first code example above. This use of concurrency allows the program to more efficiently retrieve the location details from the database.

4. Software Quality

We believe our code has internal software quality; it has been quite easy to maintain, and it is easy to read and understand. We also found that the code was easy to test and that testing it often showed us errors right away. We also think that our code has external software quality; the simplicity of the design makes it easy to use and the application runs efficiently and reliably.