Python Aptitude Test (PAT)

High Level Design Document

**Lead**

…@thundersoft.com

**Reviewers**

…@thundersoft.com

…@thundersoft.com

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.No.** | **Title** | **Version** | **Date** | **Author** | **Remarks** |
| 1. | Initial draft | V1.0 | 28/04/2021 | O.Srivalli | Initial version |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Contents**

|  |  |  |
| --- | --- | --- |
| **Sl.No.** | **Topic** | **Pg. No.** |
| 1. | Overview | 4 |
| 2. | Purpose | 4 |
| 3. | Scope | 4 |
| 4. | Requirements | 4 |
| 5. | System Requirements | 10 |
| 6. | Verification Tests | 10 |

1. **Overview:**

The Python Aptitude Test (PAT) design document mainly talks about the design specifications required to develop a web browser application for python aptitude test. This document consists of requirements and specifications used for the project of implementing a web browser app for the objective type questions on python which helps to assess fresher’s programming knowledge.

1. **Purpose:**

The main purpose of this application is to enable all freshers in TS-India, Hyd employees to take python test in order to assess their knowledge in it.

1. **Scope:**

This application is currently limited to TS-India, Hyderabad fresher’s team on python.

1. **Requirements:**

**4.1.User Requirements:**

* Formulate database (Excel) containing Python objective type questions.
* On UI, register to take the test by providing email id and password in the text boxes.
* User name should be validated first and then the credentials needs to be written to the database.
* Login with the credentials provided at registration to take the test.
* After successful verification, on next UI start test will appear.
* Then in the next UI, randomly picked 50 questions from the database will be displayed along with timer for 60 minutes.
* Also, the options for each question will also be jumbled for every user.
  1. **Functional Requirements:**
* The database i.e., PAT\_DB.xls for scripting questions on python will be totally objective type including 4 options given for each question.
* Once the app is launched, a home screen to login/register should appear to enter the credentials of the user as shown in figure 1 below.

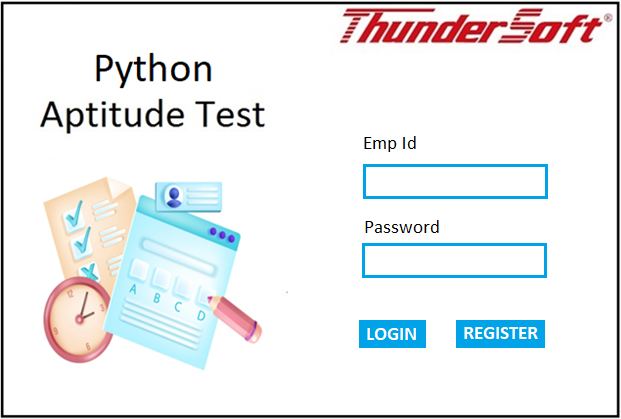


Figure 1

* Each user should first register to take the test. Upon clicking register button in the home screen, a new window should come up to accept the credentials along with a submit button as shown in Figure 2 below.
* Email id must be validated with regular expressions to accept only Thundersoft email id.

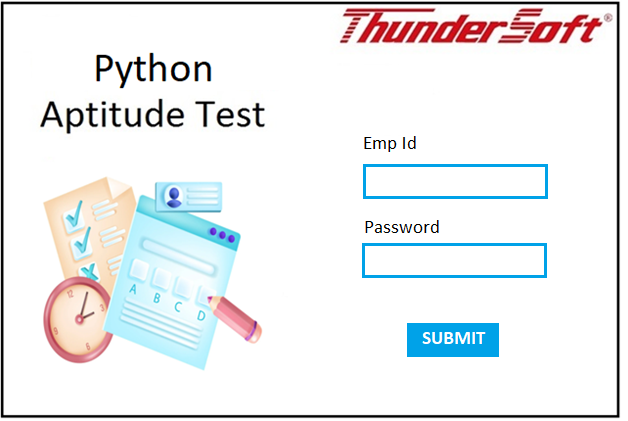


Figure 2

* Once the registration is successful, the data should be written to login sheet of PAT\_DB.xls file. Also, an Email must be triggered to the given email id.
* User can now login to take the test with the registered credentials.
* The credentials should be validated across the data provided in PAT\_DB.xls and show an error message in case the Email Id or password is incorrect.
* If login is success, it should take the user to start screen to take the test as in figure 3

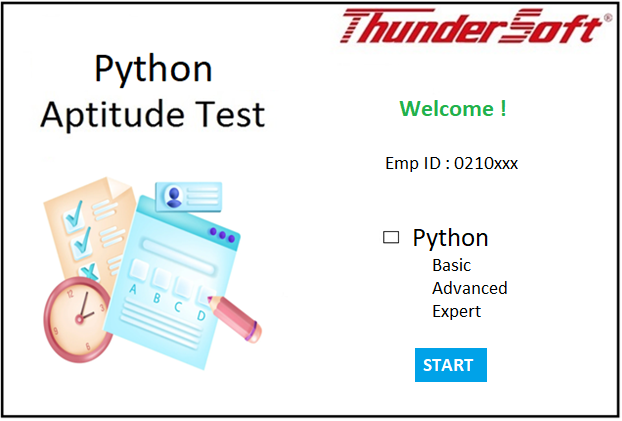


Figure 3

* Upon clicking start, user should be able to view one question at a time with following widgets.
* A text box to display the timer
* A text box to display the question number
* A test box to display the question – and a scroll bar if the length of the question exceeds the size of the text box.
* Radio buttons to display 4 options
* Answer button and Skip button

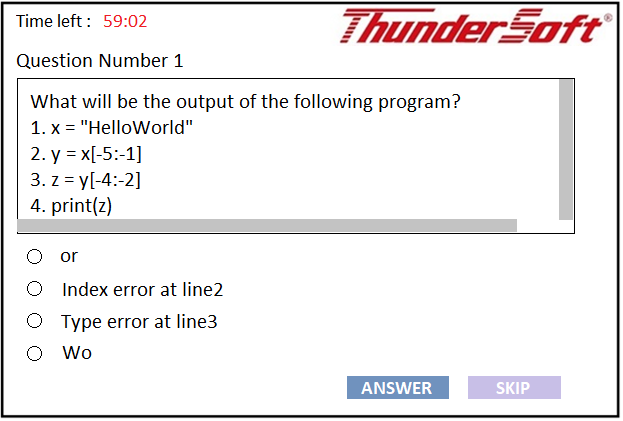
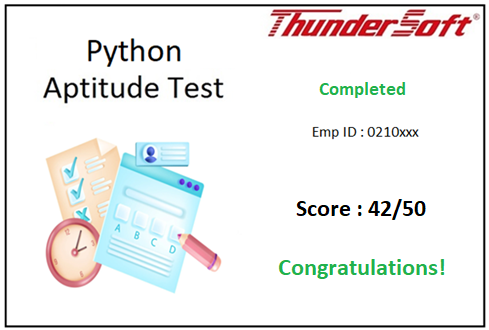


Figure 4

* When user selects an option and clicks on Answer button, user’s choice should be verified with the actual answer in the database and increment the score and display the next question.
* When user selects skip, score will not get updated but takes the user to next question.
* Once user gets a chance to answer 50 questions within the given time or when the timer shows 00:00, a screen to display the score should appear as shown in figure 5 below.
* A text box should show the message ‘Congratulations!’ if the score is greater than 40, ‘Can do better’ if the score is greater than 30 and less than 40, ‘Must improve’ if the score is less than 25.



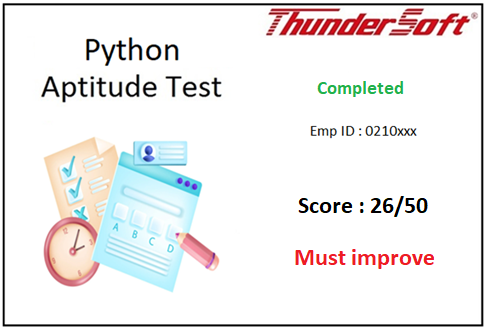
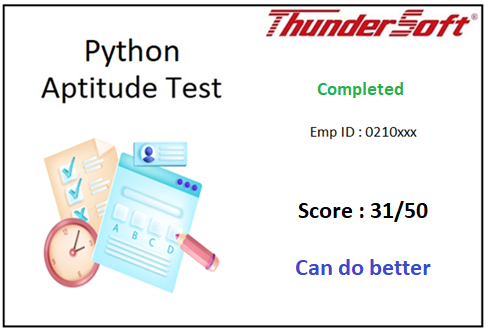


Figure 5

1. **System Requirements:**

* Language: Python <used version>
* Operating System: Works on both Windows and Linux.
  + Windows: <version> <OS details>
  + Linux: <version> <OS details>

1. **Verification Tests:**

**6.1.Test Scenario-1:**

For the user input, the questions should be randomly picked from the database and displayed with timer.

Input: Employee id

Password

Expected Output: After successful verification, randomly displayed python questions with timer.

1. **Database:**

Credentials.xlsx

|  |  |
| --- | --- |
| **Email** | **Password** |
| [abc@gmail.com](mailto:abc@gmail.com) | abc123 |
| [pqr@gmail.com](mailto:pqr@gmail.com) | pqr123 |
| [mno@gmail.com](mailto:mno@gmail.com) | mno123 |
| [xyz@gmail.com](mailto:xyz@gmail.com) | xyz123 |

Questions.xlsx

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Question** | **Option A** | **Option B** | **Option C** | **Option D** | **Answer** |
| What will be the output of the following program? 1. x = "HelloWorld" 2. y = x[-5:-1] 3. z = y[-4:-2] 4. print(z) | or | Index error at line2 | Type error at line3 | Wo | Wo |
| What will be the output of the code given below? x = 106 print(x) x = "Alfred" print(x) | Alfred | 106, Alfred | No output | Alfred, 106 | 106, Alfred |