TYPING WIZARD

AG15

8716 - Anish Nair

8723 - Hradyesh Singh

8728 - Kamran Ansari

1. Purpose of the Project

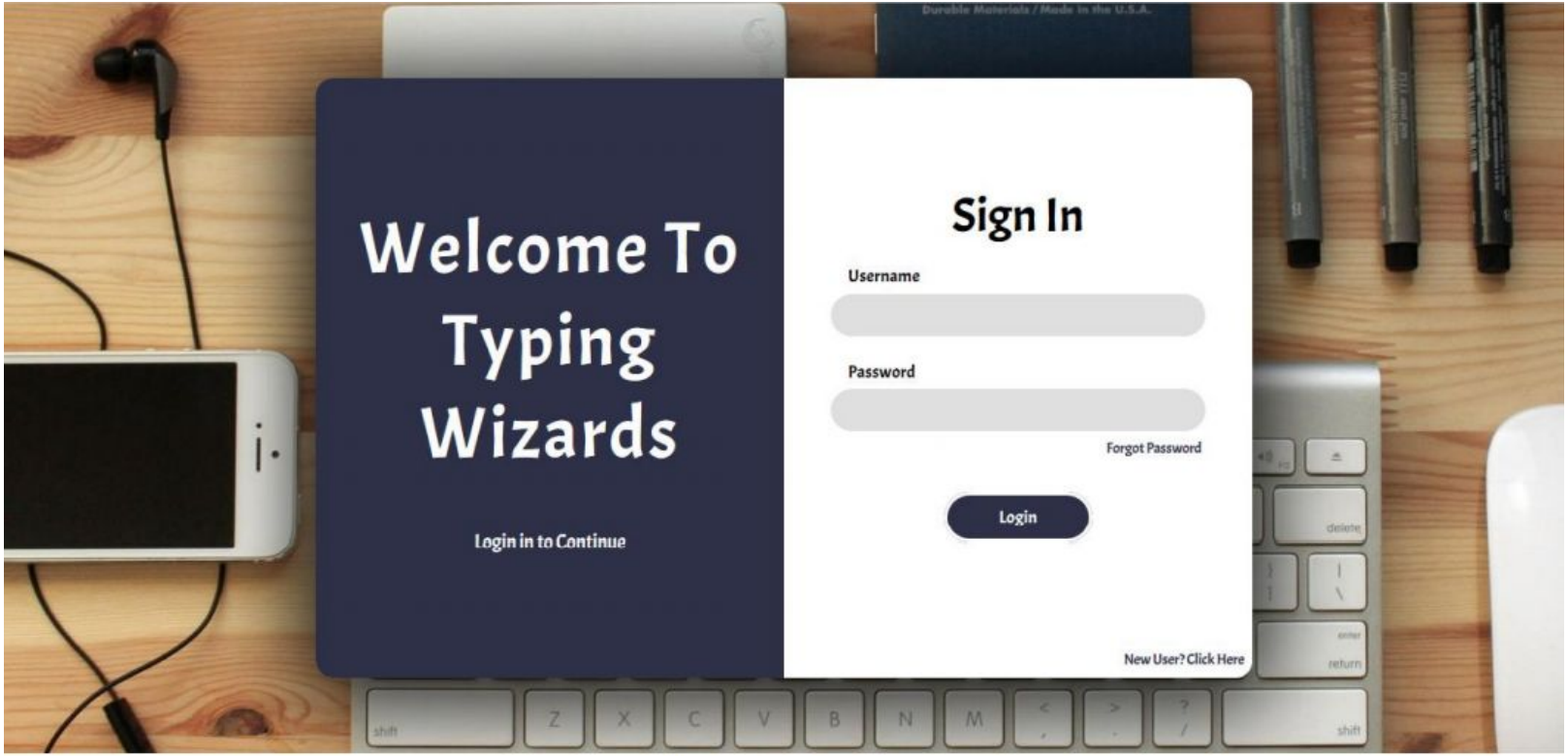
The main objective of this project is to teach Touch Typing. Touch Typing is the technique of typing without looking at the keys on the keyboard. Touch Typing increases typing speed, improves accuracy, saves time, decreases finger fatigue and reduces the risk of repetitive stress injuries. Thus overall improving the productivity of the typist.

We achieve this objective by providing the user a standard and structured typing course that teaches them touch typing at a steady pace. Goal of this tutorial course is make even a novice typist to a proficient one. The course is divided into 15 parts that each go over different sections of the keyboard. Each part introduces a set of keys that add to the cumulative set of keys introduced in the previous parts. The lessons are divided so that it becomes easy for the user to follow and learn a specific set of keys to their perfection. Each lesson has sub-lessons which provide enough practice to make the user confident in those keys. The project is made so flexible that the user can move to any lesson at any point of his training. This will help and save time for those users who are already confident about some keys.

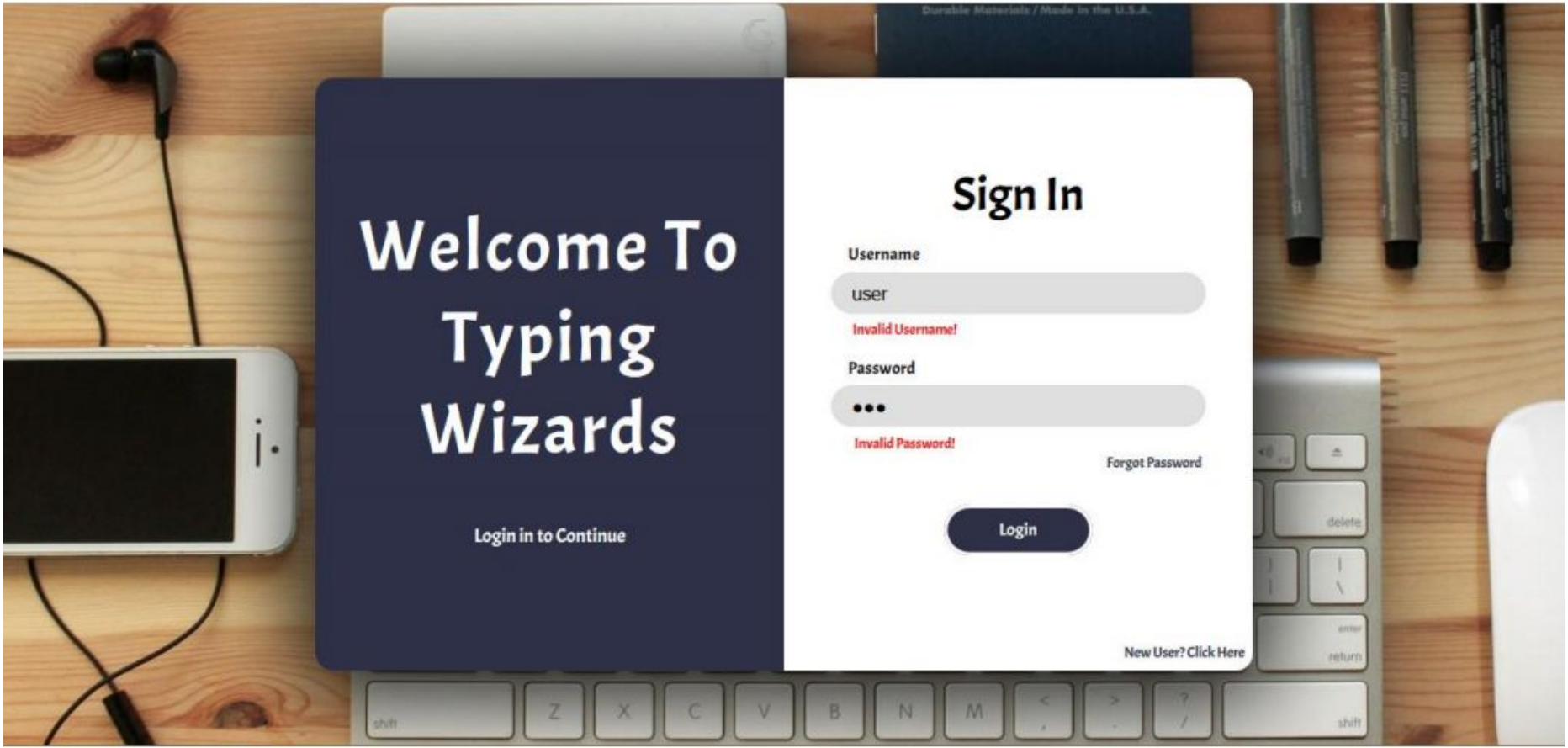
Though the project is geared towards beginners, even a seasoned touch typer can use the ‘Practice’ module to further improve their skills. Along with practicing the user can also give tests on random words generated to suit what the user has learned so far. The performance of the user in these tests is stored and used to draw graphs which give visual representation of the performance of the user. These graphs help to assess how much the user has learned so far. We also provide a ‘Game Module’ which will serve as a recreational mean for the user. Apart from this, in the ‘Challenge Module’ we have a set of paragraphs which consist of common words and phrases which one comes across everyday. This ensures that the user has mastered the basics of touch typing as he/she practices on major words that would spring up everywhere eventually.

2. Input Output Screens

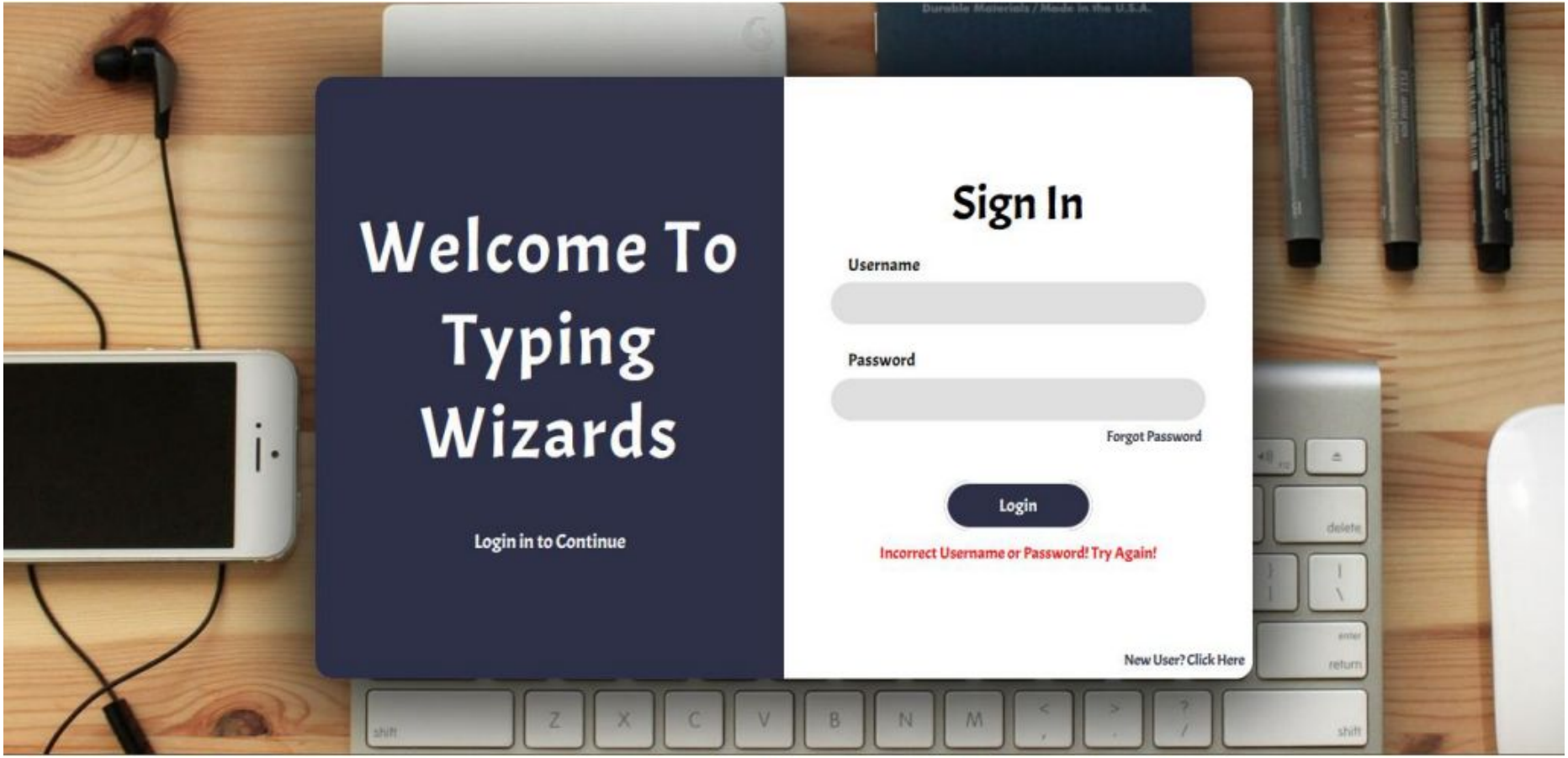
Login Screens



Login Screen



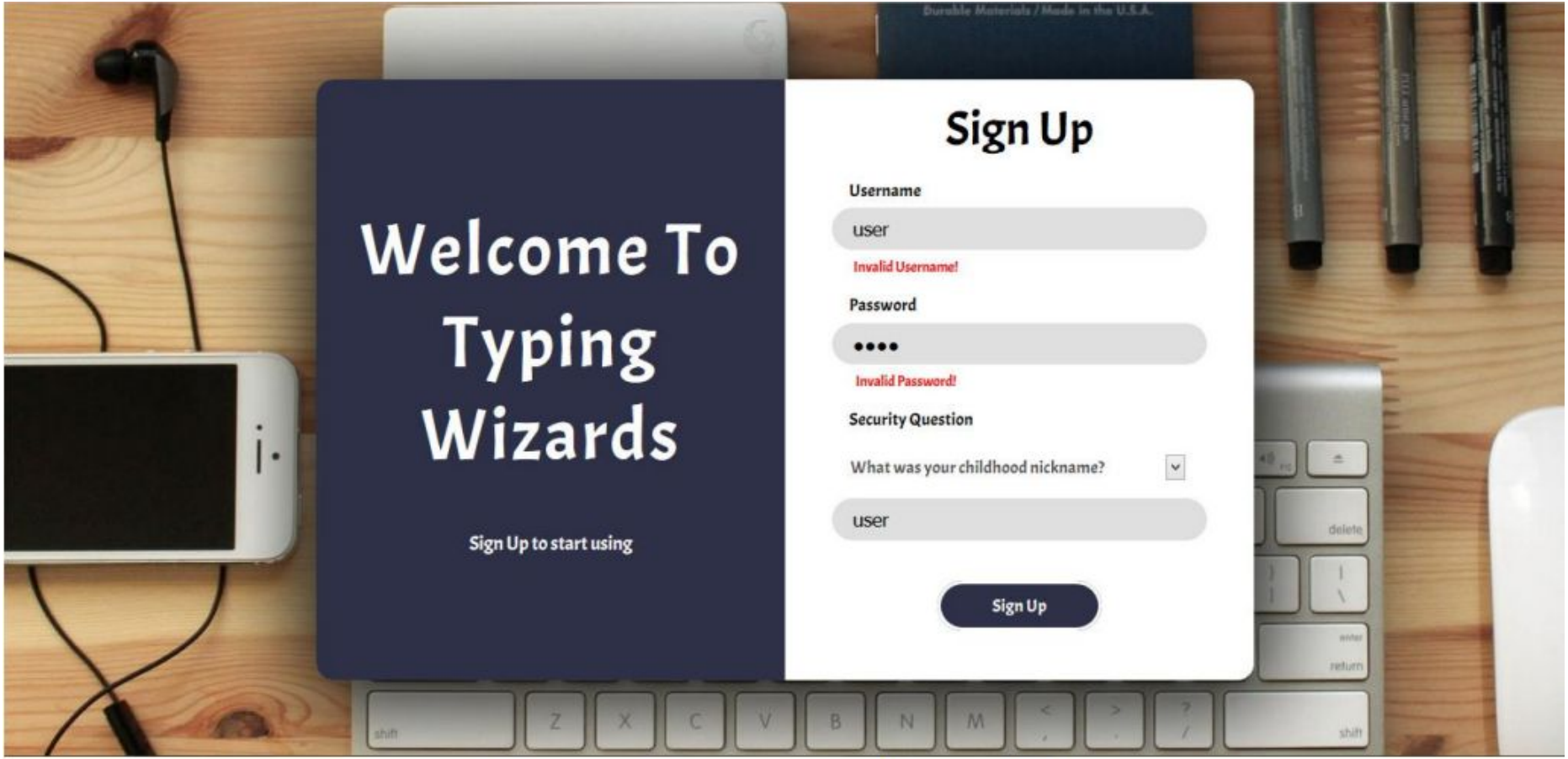
Login Screen with invalid input



Login Screen with invalid credentials

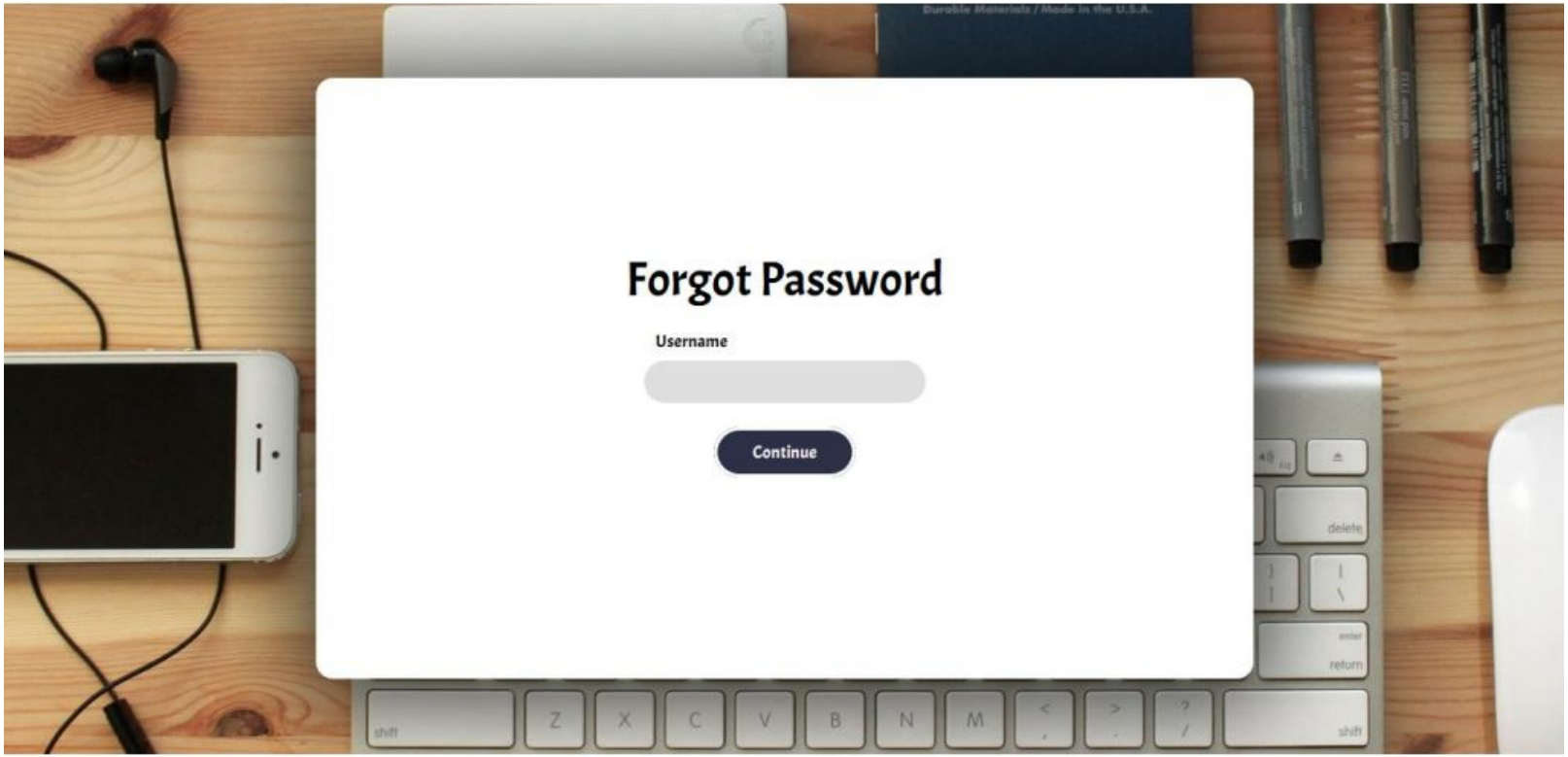
Signup Screens

Signup Screen

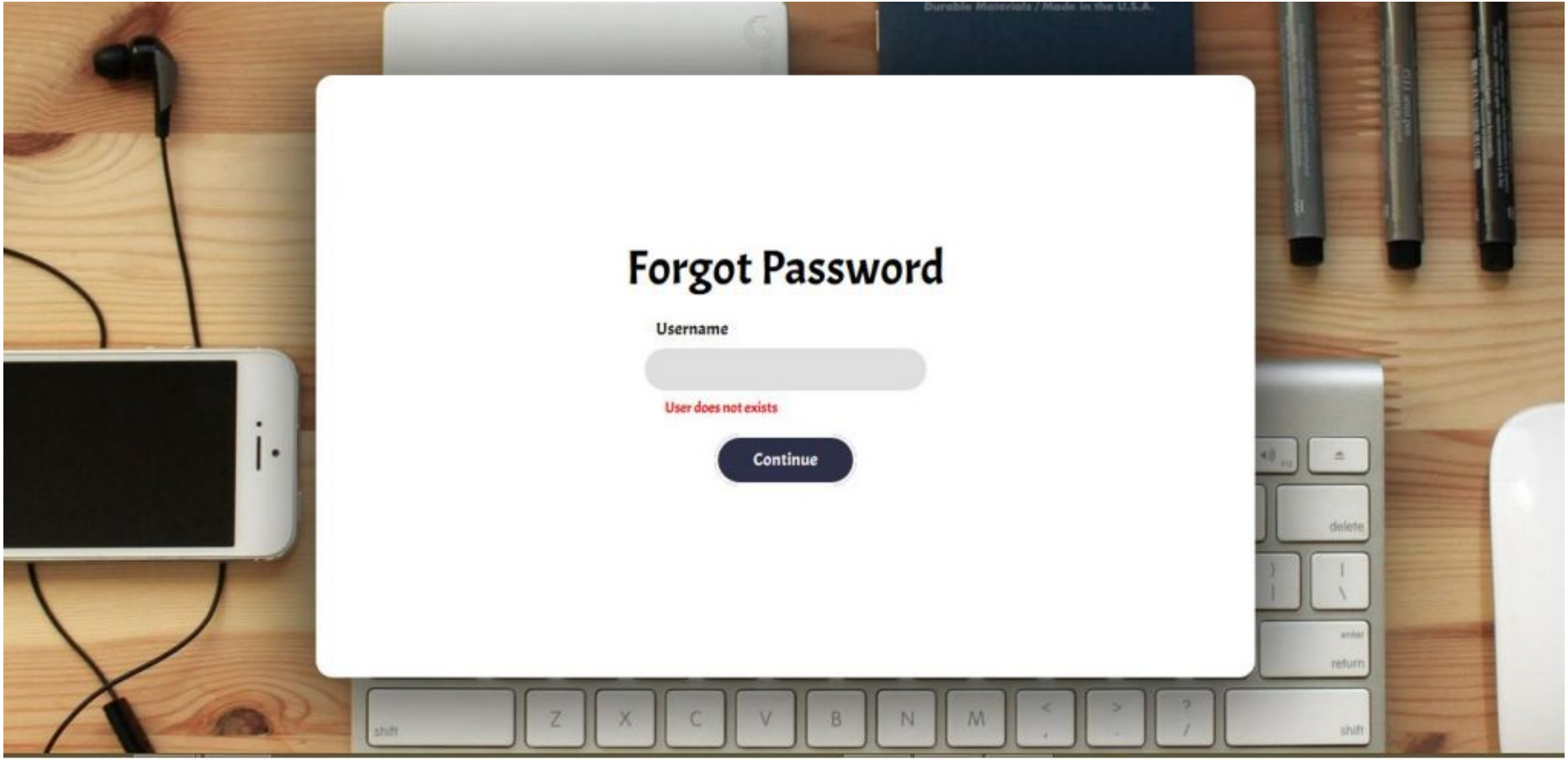


Singup Screen with invalid input

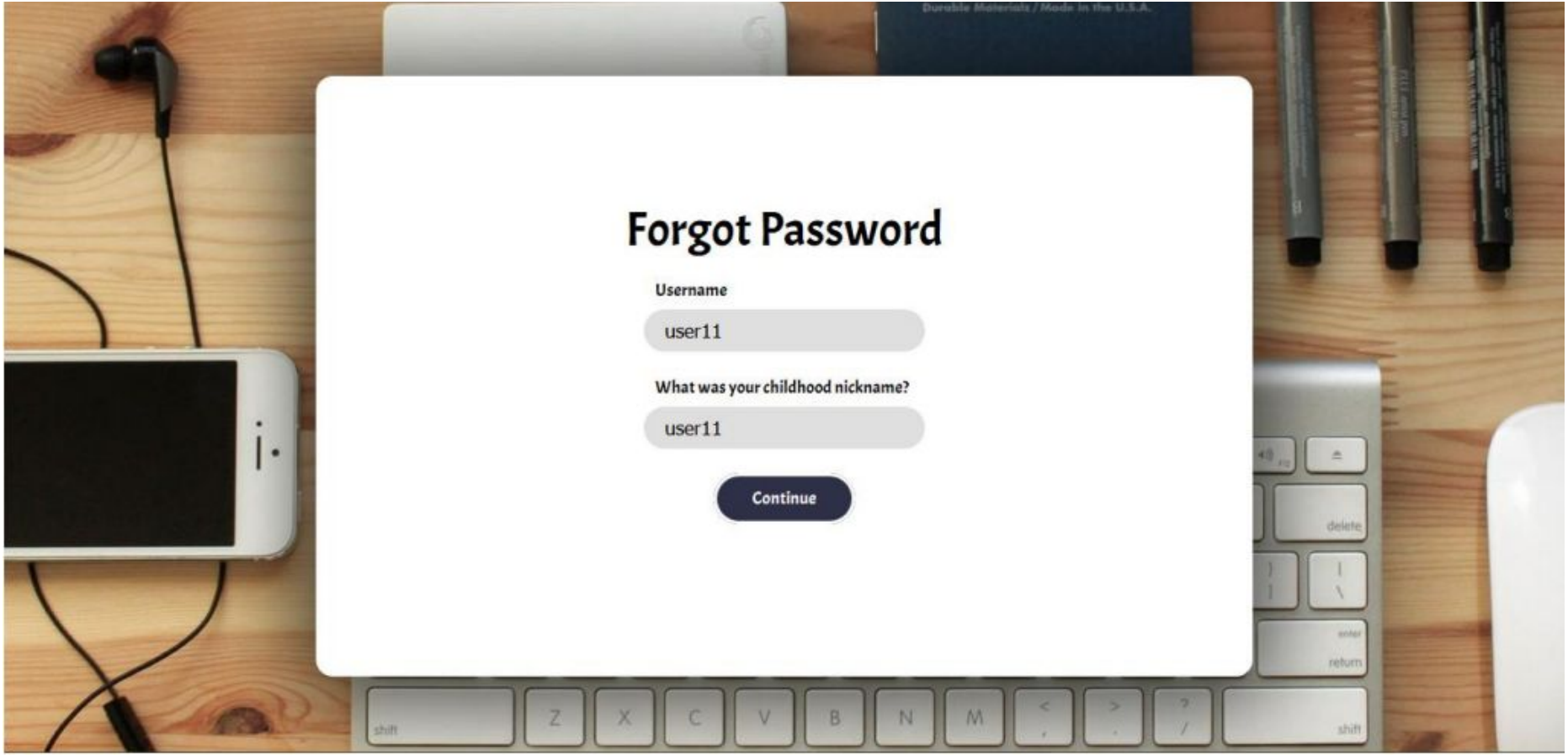
Forgot Password Screens



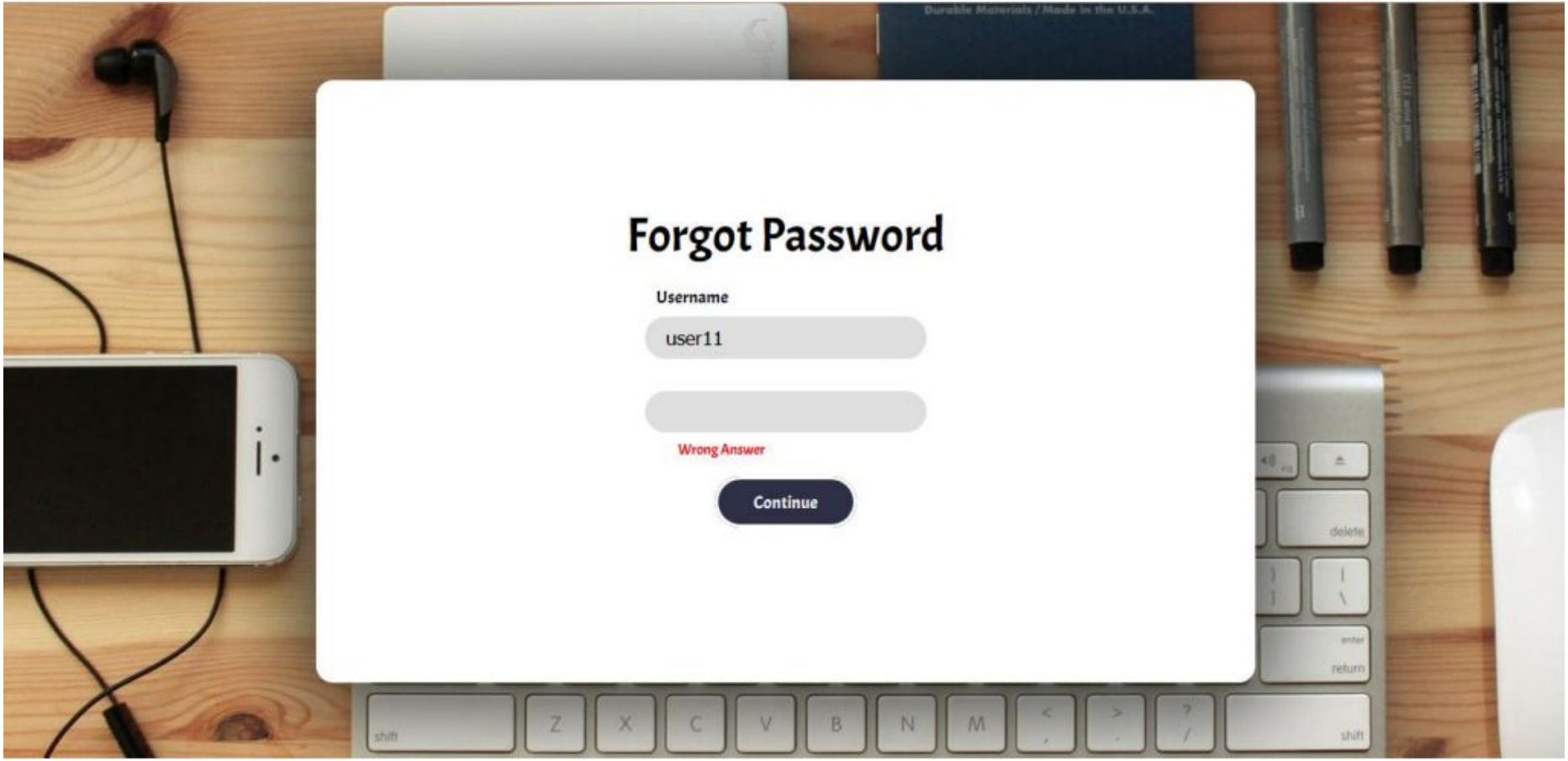
Forgot Password Screen



Forgot Password Screen invalid credentials



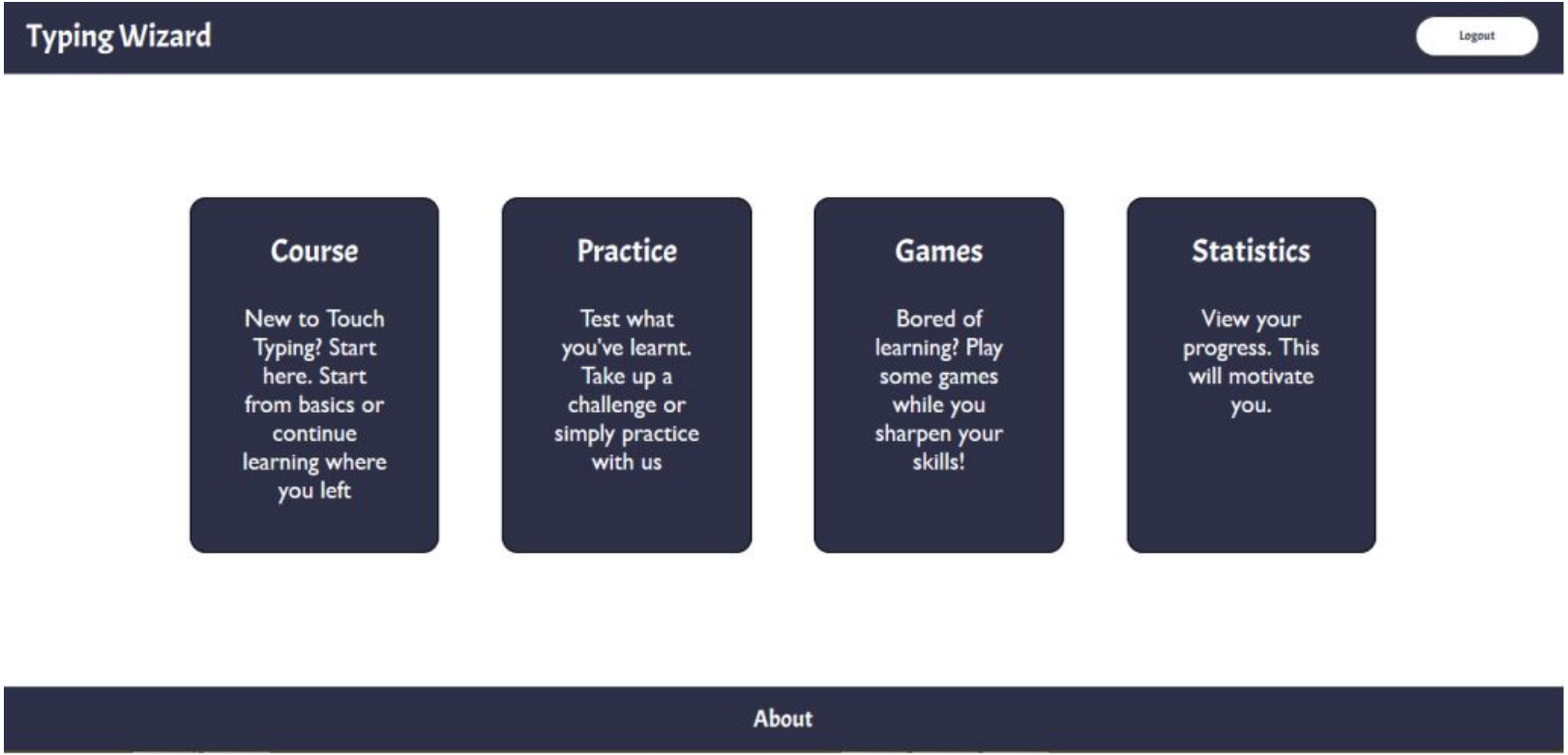
Forgot Password Screen answering question



Forgot Password Screen wrong answer

Forgot Password Screen success

Front Page Screens



Front Page Screen

Course Page Screens

Typing Wizard

Logout

Course | Practice | Games | Statistics

- Lesson 1
- Lesson 2
- Lesson 3
- Lesson 4
- Lesson 5
- Lesson 6
- Lesson 7
- Lesson 8
- Lesson 9
- Lesson 10
- Lesson 11
- Lesson 12
- Lesson 13
- Lesson 14
- Lesson 15

- New keys: w and o
- New key drill
- Key drill 1
- Key drill 2
- Word drill 1
- Word drill 2
- Word drill 3
- Blind word drill 1
- Blind word drill 2
- Text drill 1
- Text drill 2
- Extra key drill
- Extra word drill

Lesson 1

New keys: Home row

aa ss dd ff jj kk ll ;;

Time
00 : 00

Error
0

Speed
0 WPM

About

Typing Wizard

Logout

Course | Practice | Games | Statistics

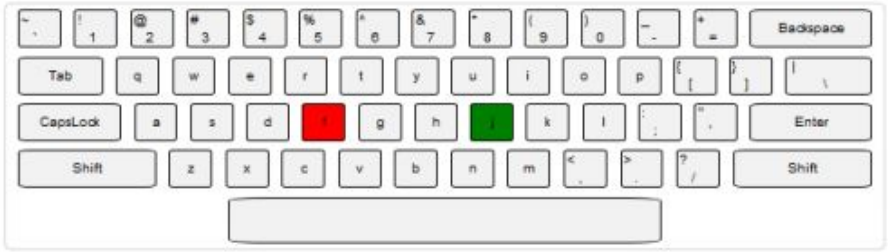
- Lesson 1
- Lesson 2
- Lesson 3
- Lesson 4
- Lesson 5
- Lesson 6
- Lesson 7
- Lesson 8
- Lesson 9
- Lesson 10
- Lesson 11
- Lesson 12
- Lesson 13
- Lesson 14
- Lesson 15

Lesson 1

New keys: Home row

aa ss dd ff jj kk ll ;;

aa ss dd ff f



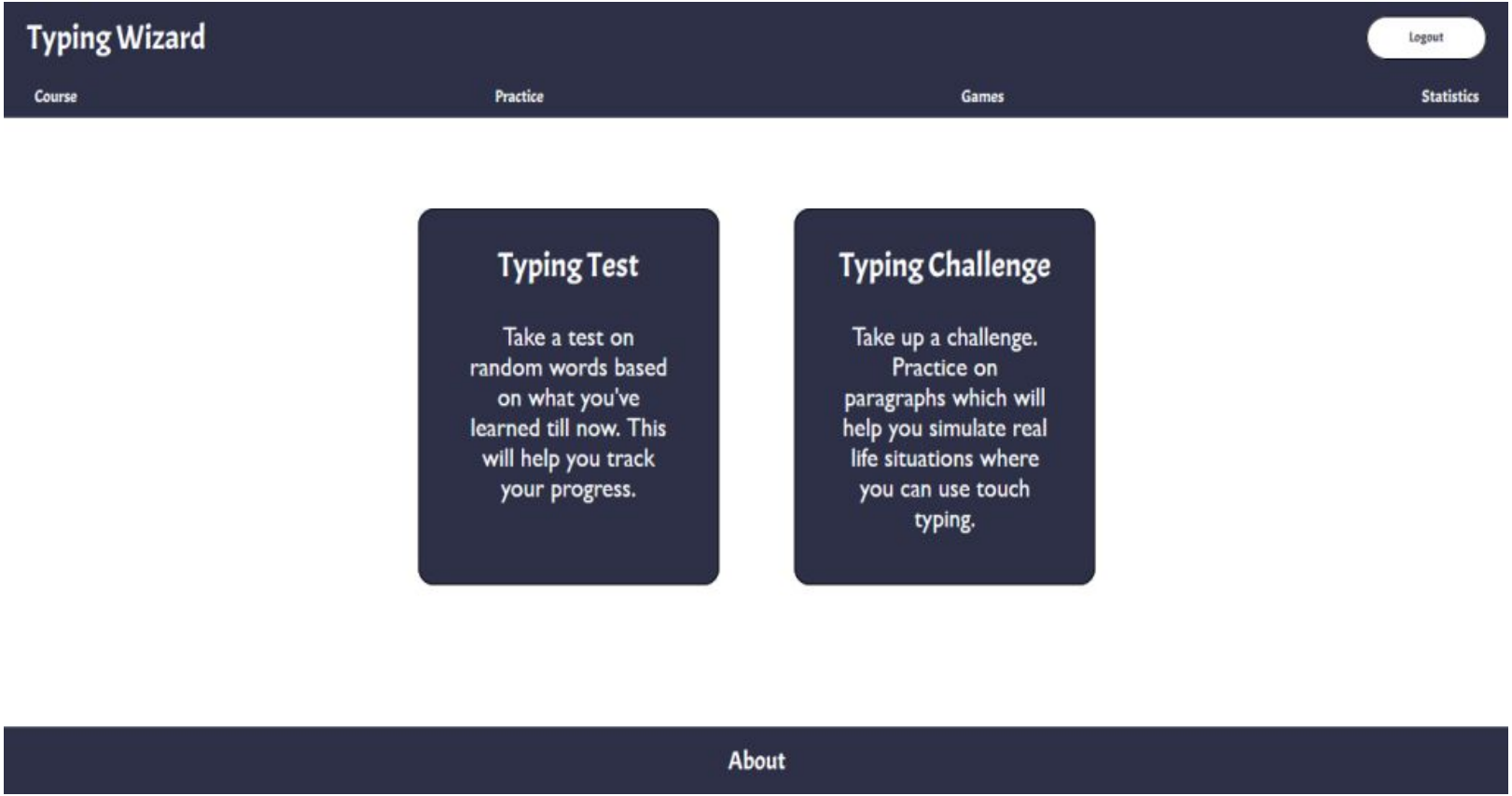
Time
00 : 22

Error
1

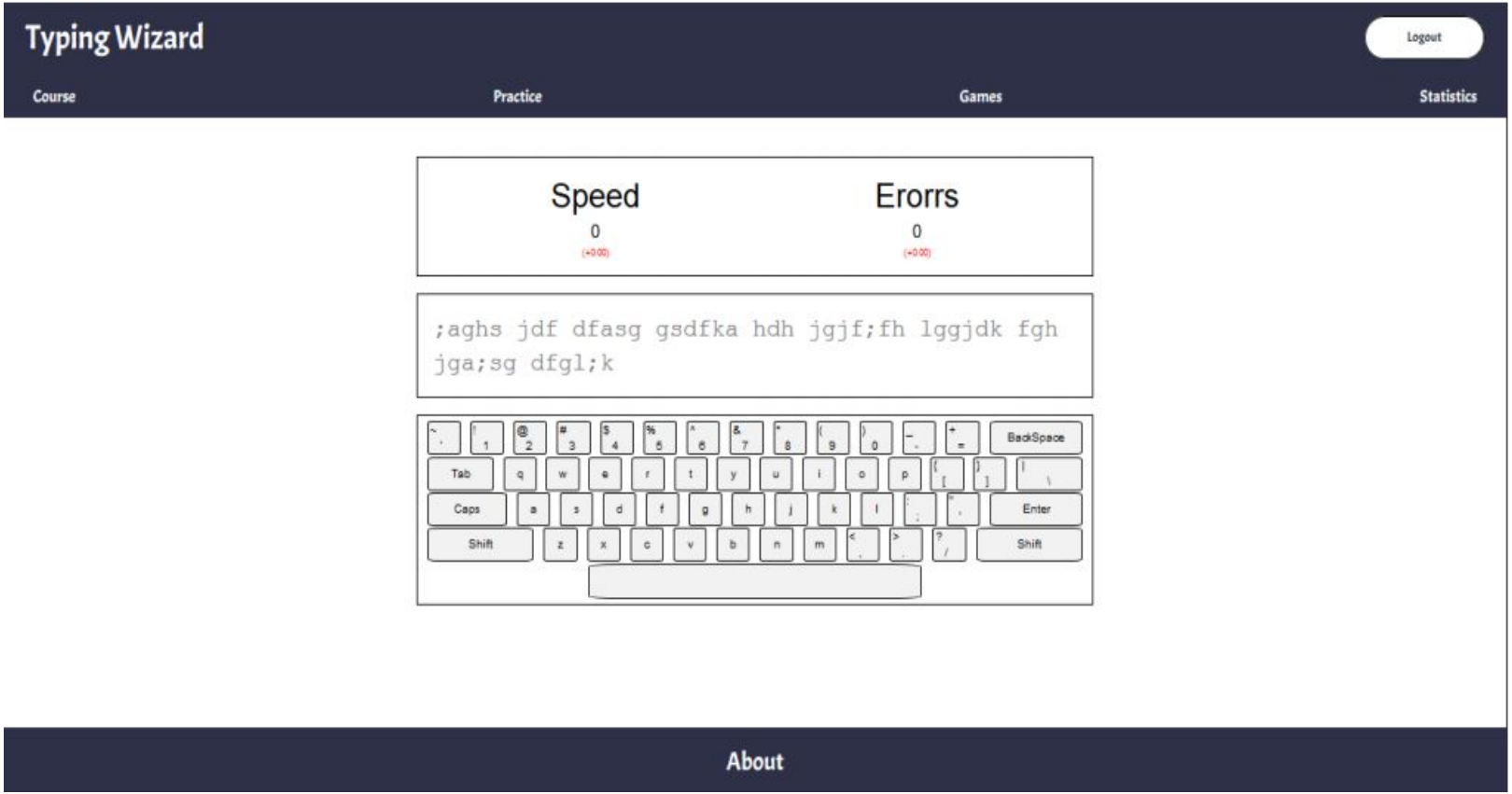
Speed
7WPM

About

Practice Page Screens



Typing Test Screens



Typing Wizard

Logout

Course | Practice | Games | Statistics

Speed	Erorrs
43.72 (+1.68)	4 (+1.00)

gakl d;adghk d;ah; lghkfd afgl lhdh las ghshlfg l
hjs jg;

About

Typing Wizard

Logout

Course | Practice | Games | Statistics

Speed	Erorrs
40.20 (-3.68)	4 (+2.00)

kfdgf afkdl ajsgk dahjag hlsjl ;khakj kdhs s;skgf f agj;d j;lgd

About

Typing Challenge Screens

Typing Challenge

The Wolf and the Lamb

A Wolf came upon a Lamb straying from the flock, and felt some compunction about taking the life of so helpless a creature without some plausible excuse; so he cast about for a grievance and said at last, "Last year, sirrah, you grossly insulted me." "That is impossible, sir," bleated the Lamb, "for I wasn't born then." "Well," retorted the Wolf, "you feed in my pastures." "That cannot be," replied the Lamb, "for I have never yet tasted grass." "You drink from my spring, then," continued the Wolf. "Indeed, sir," said the poor Lamb, "I have never yet drunk anything but my mother's milk." "Well, anyhow," said the Wolf, "I'm not going without my dinner"; and he sprang upon the Lamb and devoured it ithout more ado

The Lion and the Mouse
The Fox and the Grapes
The Charcoal-Burner and the Fuller
The Goose that Laid the Golden Eggs
The Cat and the Mice
The Mischievous Dog
The Mice in Council
The Bat and the Weasels
The Dog and the Sow
The Fox and the Crow
The Horse and the Groom
The Wolf and the Lamb

The Wolf and the Lamb

Typing Challenge

The Wolf and the Lamb

A Wolf came upon a Lamb straying from the flock, and felt some compunction about taking the life of so helpless a creature without some plausible excuse; so he cast about for a grievance and said at last, "Last year, sirrah, you grossly insulted me." "That is impossible, sir," bleated the Lamb, "for I wasn't born then." "Well," retorted the Wolf, "you feed in my pastures." "That cannot be," replied the Lamb, "for I have never yet tasted grass." "You drink from my spring, then," continued the Wolf. "Indeed, sir," said the poor Lamb, "I have never yet drunk anything but my mother's milk." "Well, anyhow," said the Wolf, "I'm not going without my dinner": and he sprang upon the Lamb and devoured it ithout more ado

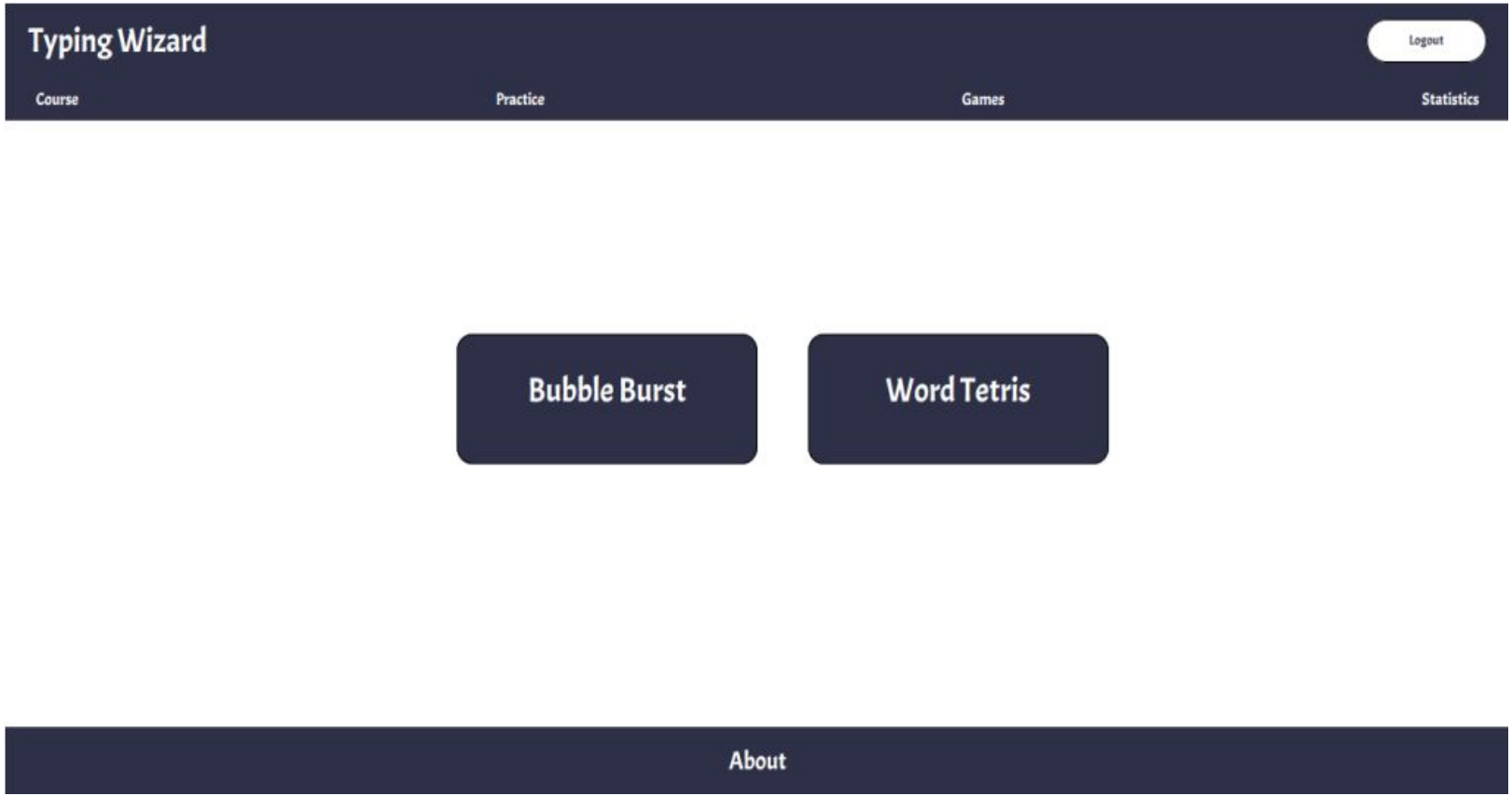
A Wolf came upon a Lamb sgraying from the flock, and felt

Choose another story

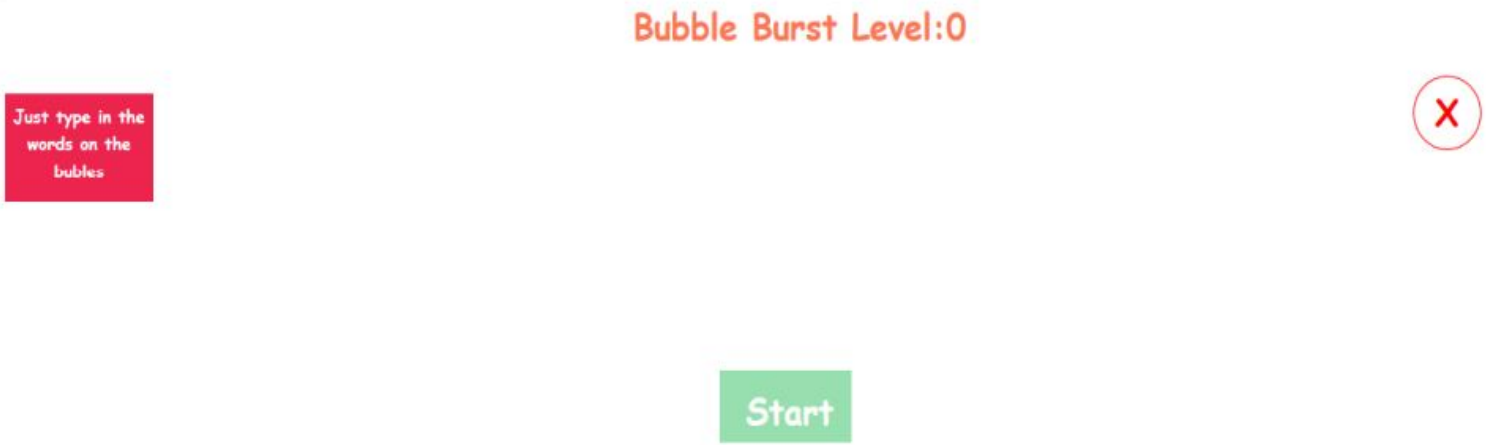
The Wolf and the Lamb


Typing Wizard
Logout
Course
Practice
Games
Statistics
Typing Challenge
The Fox and the Grapes
A hungry Fox saw some fine bunches of Grapes hanging from a
trained along a high trellis, and did his best to reach them
high as he could into the air. But it was all in vain, for t
out of reach: so he gave up trying, and walked away with an
and unconcern, remarking, "I thought those Grapes were ripe,
they are quite sour.
aw some fine bunches of Grapes hanging from a vine that was
a high trellis, and did his best to reach them by jumping as
ld into the air. But it was all in vain, for they were just
so he gave up trying, and walked away with an air of dignity
remarking, "I thought those Grapes were ripe, but I see now
sour,.
x
Time Taken
1 Minutes 23 Seconds
Words Per Minute
64.03
Errors
1
Choose another story
The Fox and the Grapes
About

Game Screens



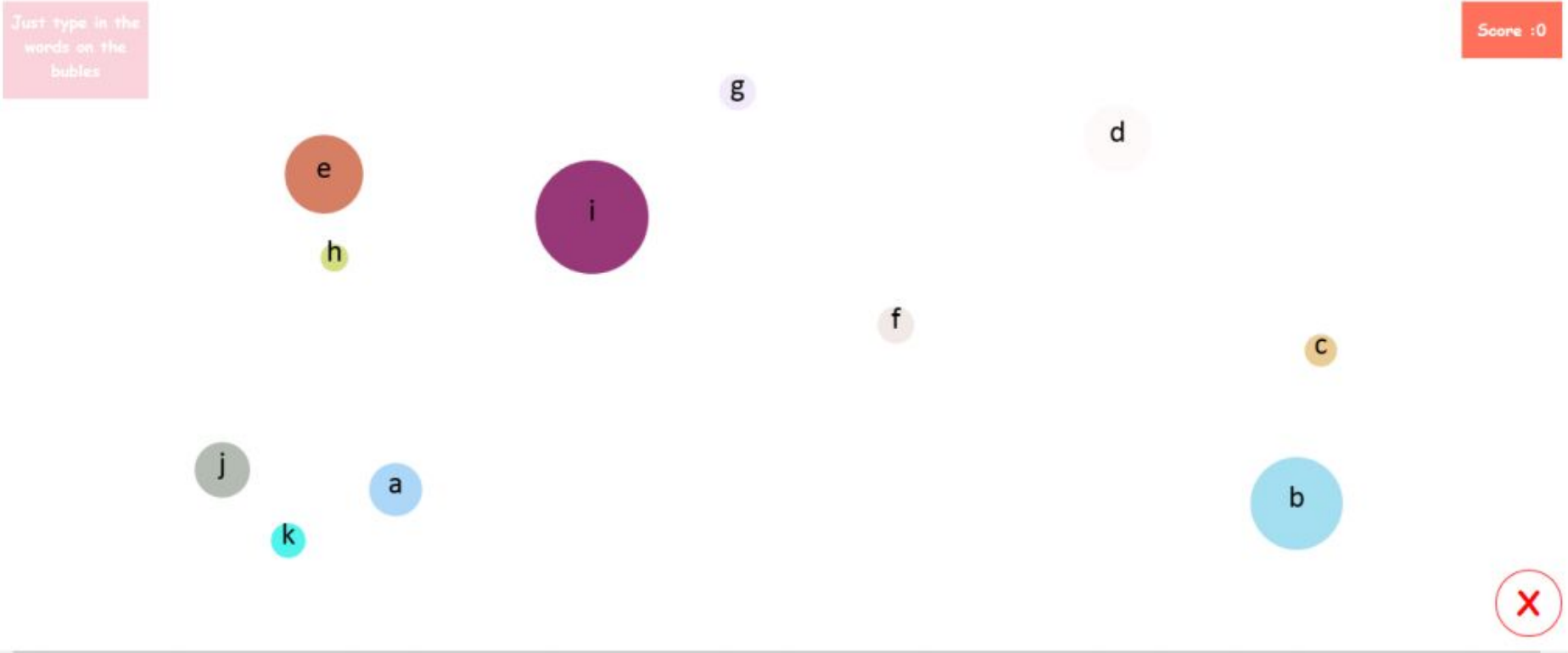
Bubble Burst Screens



Bubble Burst Level:1

Just type in the words on the bubles

Score :0



Bubble Burst Level:1

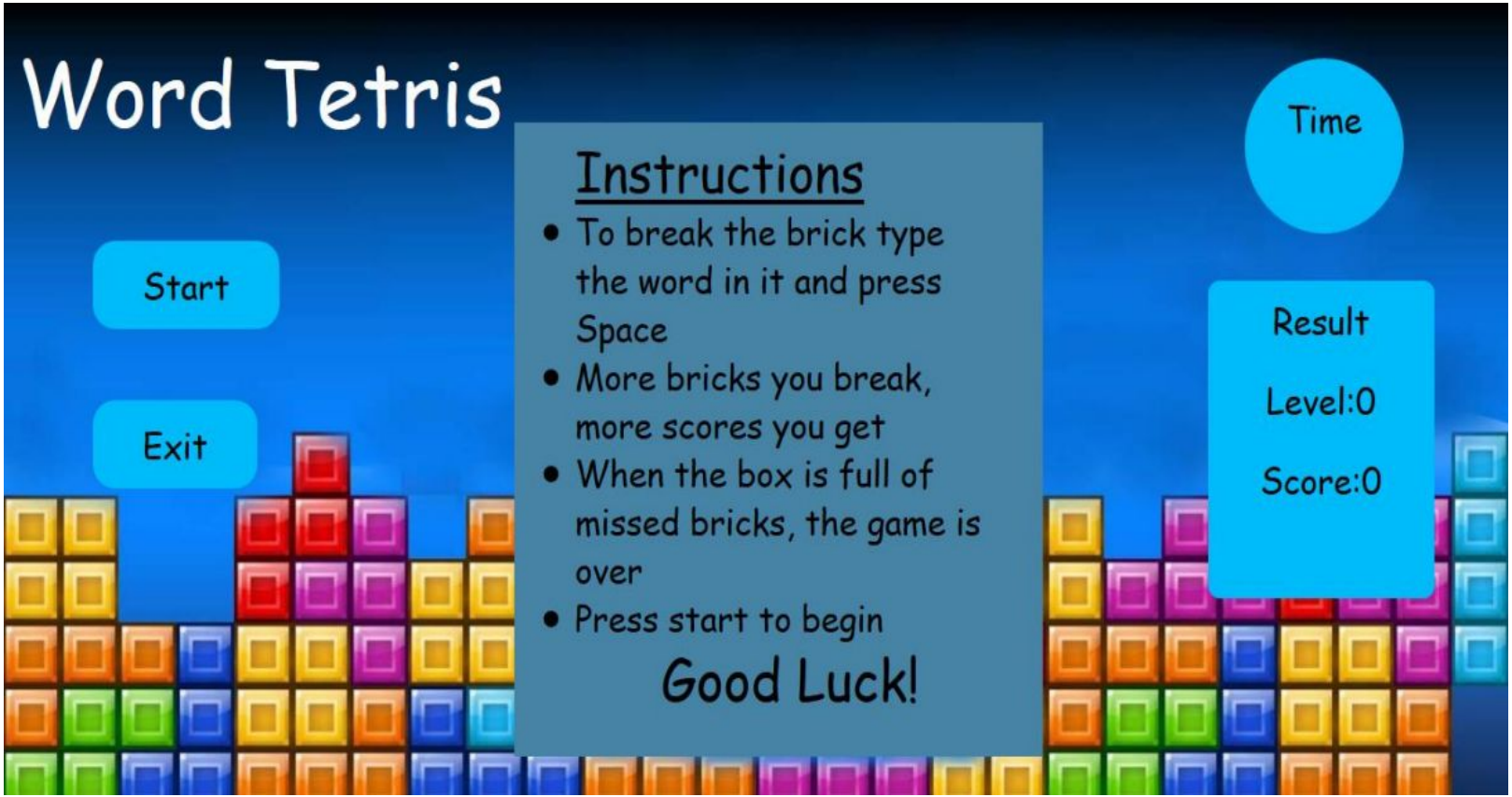
Just type in the words on the bubles

Time Taken: 51.615

You could'nt clear this round,
TRY AGAIN!!!

Start

Word Tetris Screens

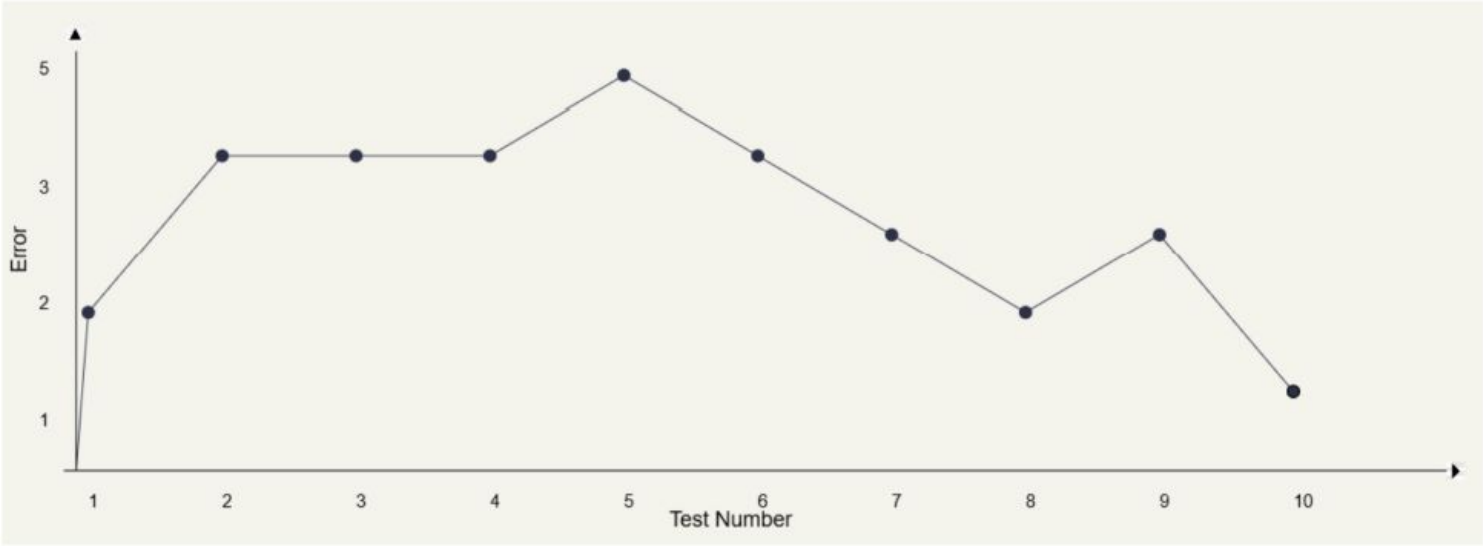



Word Tetris
Time
00 : 07
Start
Exit
MILE
Result
Level:1
Score:0


Word Tetris
Time
00 : 36
Start
STRIP
Exit
Result
Level:1
Score:0
GOLDEN
MILE

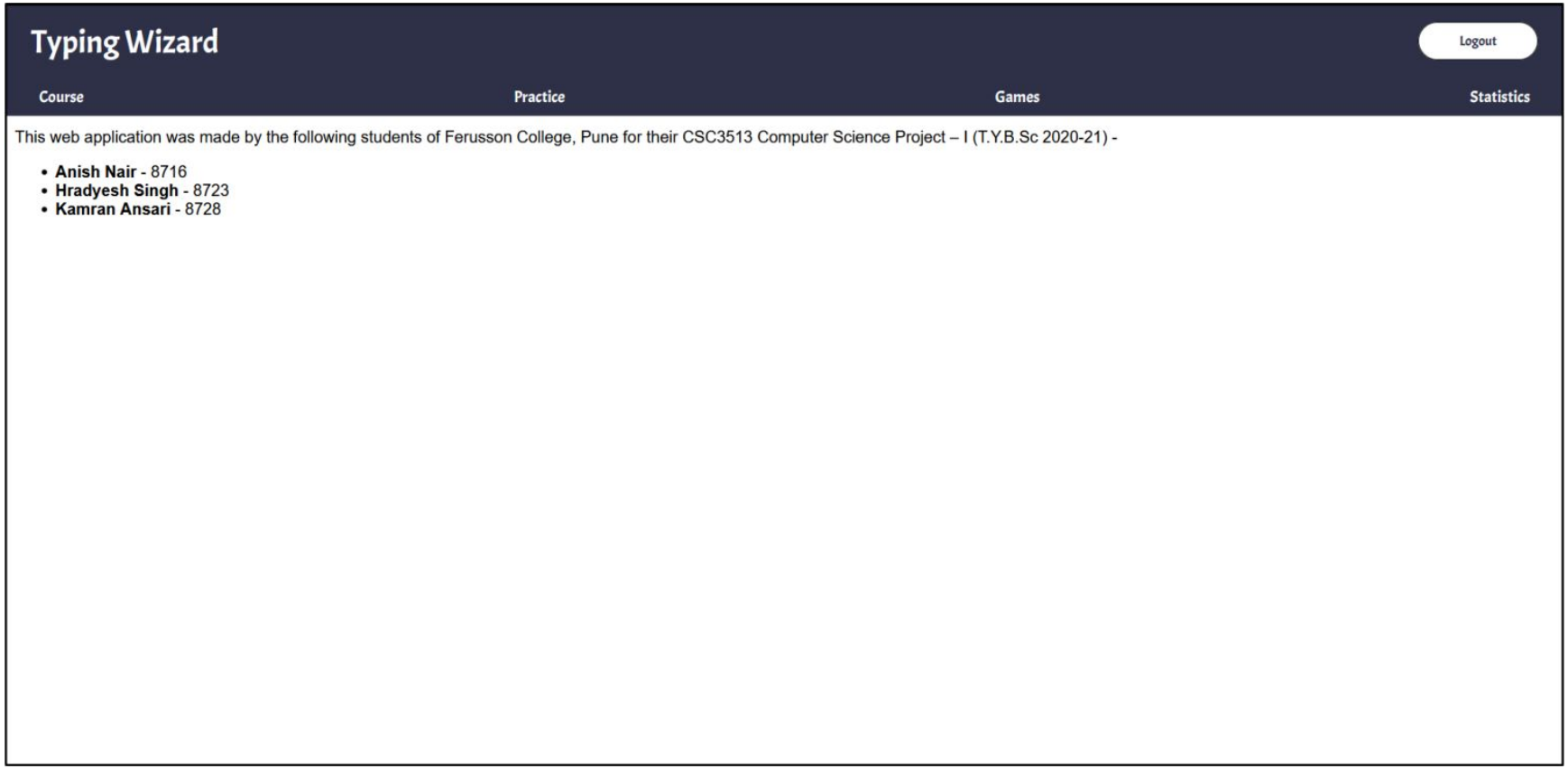

Word Tetris
Game Over
Press start to play again
Time
01 : 03
Start
Exit
ROSE
TEMPERATURE
STRIP
GOLDEN
MILE
Result
Level:1
Score:0

Statistics Screens



Test Number vs Error

About Screens



3. Reports Generation

Signup

The SignUp page takes form input from the user and stores the login information in the **`users`** table.

Description of **`users`** table -

```
      Column      |          Type          | Collation | Nullable | Default
------------------+------------------------+-----------+----------+---------
 username         | character varying(100) |           | not null |
 password         | character varying(100) |           | not null |
 forgotquestionno | integer                |           |          |
 answer           | text                   |           | not null |
Indexes:
    "users_pkey" PRIMARY KEY, btree (username)
Foreign-key constraints:
    "users_forgotquestionno_fkey" FOREIGN KEY (forgotquestionno) REFERENCES forgotpassquestions(qno) ON UPDATE SET NULL ON DELETE CASCA
DE
Referenced by:
    TABLE "lessonscompleted" CONSTRAINT "lessonscompleted_username_fkey" FOREIGN KEY (username) REFERENCES users(username) ON UPDATE SE
T NULL ON DELETE CASCADE
    TABLE "typingtestuser" CONSTRAINT "typingtestuser_username_fkey" FOREIGN KEY (username) REFERENCES users(username) ON UPDATE SET NU
LL ON DELETE CASCADE
    TABLE "userstats" CONSTRAINT "userstats_username_fkey" FOREIGN KEY (username) REFERENCES users(username) ON UPDATE SET NULL ON DELE
TE CASCADE
```

Entered form input is validated on both client and server side. Server side validations fetch data from the database with SQL queries -

1. Check if username already exists.

```
SELECT * from users where username='{$userName}'
```

Where, `userName` is username entered by the user in the signup form.

On successful registration the user data is stored in the **`users`** table-

username	password	forgotquestionno	answer
userss	password	1	userss
shivam	123456	1	kamran
user11	user11	1	user11
vishal	1234567	1	vishal
User11	pass11	1	user11
popopo	popopo	1	popopo
hgcgv_bdsvu	12384655svgbu	1	ufuy ddsv
Rakshita	rakshitasingh	2	subway surfers
meep2001	meep2001	1	meep
user456	user456	1	koko
user45	user45	2	khokho
usersss	123456789	1	kk
usersk	123456	1	ss
usersps	123456	1	kk
user12	asdfjklj	2	khokho
kussssh29	123456	1	kamran

(16 rows)

Login

The Login page authenticates the user using the submitted form input.

Login also uses **`users`** table to authenticate users.

Login form is also validated both on client and server side. Server side validations fetch data from the database with SQL queries -

1. Check if username exists.

   ```
   SELECT * from users where username='{$userName}'
   ```

   Where, `userName` is username entered by the user in the login form.

2. Check if entered password is correct.

   ```
   SELECT * from users where username='{$userName}'
   ```

   Password is accessed as `$res["password"]` where `res` is the result associated array of this query.

Results from these queries are also used to show respective errors to the user.

On a successful authentication a cookie is stored on the client machine which is used to validate the user's session.

Name	Value	Domain	Path	Expires / Max-Age	Size
username	userss	typing-wizar...	/	Mon, 31 May 2021...	14

This cookie's expiration is set to 1 day.

Logout

On logout this cookie is deleted from the user's machine.

Forgot Password

Using forgot password functionality the user can retrieve their forgotten password from the database using tables **`forgotpassquestions`** and **`users.`**

Description of **`forgotpassquestions`** table -

```
 Column  |  Type   | Collation | Nullable | Default
---------+---------+-----------+----------+---------
 qno     | integer |           | not null |
 question | text   |           |          |
Indexes:
    "forgotpassquestions_pkey" PRIMARY KEY, btree (qno)
Referenced by:
    TABLE "users" CONSTRAINT "users_forgotquestionno_fkey" FOREIGN KEY (forgotquestionno) REFERENCES forgotpassquestions(qno) ON UPDATE
 SET NULL ON DELETE CASCADE
```

Following queries are used to fetch the required data from the database-

1. Check if username exists and if exists get their security question number.

   ```
   SELECT * from users where username='{$userName}'
   ```

   Where, `userName` is username entered by the user in the login form. Question number is accessed as `$res['forgotquestionno']` where `res` is result associated array of this query.

2. Fetch user's security question text

```
SELECT * FROM forgotpassquestions WHERE qno=$no
```

Where, `no` is security question number obtained from the last query. `$res['question']` is used to get the question text. `res` is the result associated array of this query.

```
 qno |                 question
-----+----------------------------------
   1 | What was your childhood nickname?
   2 | What is your favorite game?
   3 | Who is your childhood sports hero?
(3 rows)
```

3. Fetch correct answer and the password from the user's table

```
SELECT * from users where username='{$userName}'
```

Answer is accessed as `$res['answer']` and password as `$res['password']` where `res` is result associated array of this query.

Results from these queries are also used to show respective errors to the user.

Course/Lesson

The required data is obtained from the database using XHR and tables **`lessons`** and **`lessonscompleted`**.

Description of **`lessons`** -

```
   Column   |          Type          | Collation | Nullable | Default
------------+------------------------+-----------+----------+---------
 lessonid   | character varying(10)  |           | not null |
 mockpara   | text                   |           |          |
 lessonname | character varying(100) |           |          |
Indexes:
    "lessons_pkey" PRIMARY KEY, btree (lessonid)
Referenced by:
    TABLE "lessonscompleted" CONSTRAINT "lessonscompleted_lessonid_fkey" FOREIGN KEY (lessonid) REFERENCES lessons(lessonid) ON UPDATE
SET NULL ON DELETE CASCADE
```

Description of **`lessonscompleted`** -

```
  Column  |          Type          | Collation | Nullable | Default
----------+------------------------+-----------+----------+---------
 username | character varying(100) |           | not null |
 lessonid | character varying(10)  |           | not null |
Indexes:
    "lessonscompleted_pkey" PRIMARY KEY, btree (username, lessonid)
Foreign-key constraints:
    "lessonscompleted_lessonid_fkey" FOREIGN KEY (lessonid) REFERENCES lessons(lessonid) ON UPDATE SET NULL ON DELETE CASCADE
    "lessonscompleted_username_fkey" FOREIGN KEY (username) REFERENCES users(username) ON UPDATE SET NULL ON DELETE CASCADE
```

1. Fetch sub lesson list (getSubLesson.php)

   This XHR request is made whenever the user hovers over the lesson list present of the left side of the course page. It builds and returns the HTML of the list consisting of the sub lessons of the passed lessonid. The SQL query used is -

   ```
   SELECT * FROM lessons WHERE lessonid LIKE '{$lessonNo}.{$sublesson}%'
   ```

   Where, `lessonNo` is the lessonid provided on XHR request. And `sublesson` iterates from 1 to maximum possible sublesson number.

2. Fetch latest completed lesson's id (getLessonsCompleted.php)

   This request is made on the page load to get the last completed lesson.

   ```
   SELECT lessonid from lessonsCompleted where username='{$username}'
   SELECT count(lessonid) from lessonsCompleted where username='{$username}'
   SELECT count(lessonid) from lessons
   ```

   Returns array of last completed lessonid, count of total completed lessons and count of all lessons in the lessons table.

3. Fetch lesson content (getLessons.php)

```
SELECT * FROM lessons WHERE lessonid LIKE '".$lessonid.".%'
```

Result of query will be like this -

```
 lessonid |        mockpara          |     lessonname
----------+--------------------------+--------------------
 1.1.1    | aa ss dd ff jj kk ll ;;  | New keys: Home row
 1.1.2    | as as df df jk jk l; l;  | New keys: Home row
 1.1.3    | as as df df jk jk l; l;  | New keys: Home row
 1.1.4    | asdf asdf jkl; jkl;      | New keys: Home row
 1.1.5    | fdsa fdsa ;lkj ;lkj      | New keys: Home row
 1.1.6    | jkl; jkl; asdf asdf      | New keys: Home row
 1.1.7    | ;lkj ;lkj fdsa fdsa      | New keys: Home row
(7 rows)
```

4. Fetch lesson that was not completed (getMinLessonsNotCompleted.php)

This XHR request is triggered when the user finishes the last lesson of the course. It fetches the minimum lessonid that was skipped over/not completed by the user.

```
SELECT lessonid from lessonsCompleted where username='{$username}'
```

```
SELECT lessonid from lessons
```

5. Insert completed lessonid into the database (putLessonsCompleted.php)

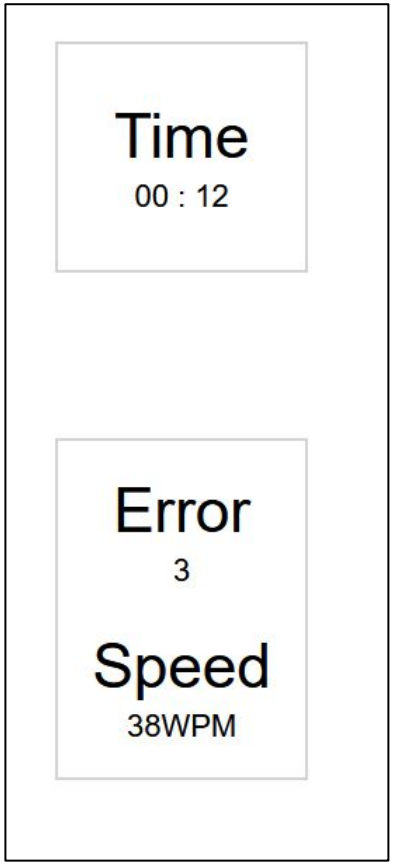
This request is made when the user completes a sub sub lesson.

```
INSERT into lessonscompleted values('{$username}','{$Id}')
```

Where, `Id` is the completed sub sub lessonid.

username	lessonid
userss	1.1.1
userss	1.1.2
userss	1.1.3
userss	1.1.4
userss	1.1.5
userss	1.1.6
userss	1.1.7
userss	1.2.1
userss	1.2.2
userss	1.2.3
userss	4.1.1

As the user completes the lessons, a panel on the right hand side of the page displays user statistics simultaneously.



Typing Test

The user can practice on randomly generated paragraphs. Paragraphs are generated using only those keys that the user have learned up until that point from the lesson module. The generated statistical data is sent to be stored in the database. A table displays relative performance of the last test completed. The data is acquired and posted through XHR requests and the tables used are **`lessonscompleted`**, **`userstats`** and **`typingtestuser`**

Description of **`userstats`** -

```
    Column    |          Type          | Collation | Nullable | Default
--------------+------------------------+-----------+----------+---------
 username     | character varying(100) |           |          |
 topspeed     | double precision       |           |          |
 averagespeed | double precision       |           |          |
 averageerror | double precision       |           |          |
 totalsamples | integer                |           |          |
Foreign-key constraints:
    "userstats_username_fkey" FOREIGN KEY (username) REFERENCES users(username) ON UPDATE SET NULL ON DELETE CASCADE
```

Description of **`typingtestuser`** -

```
  Column  |          Type          | Collation | Nullable | Default
----------+------------------------+-----------+----------+---------
 testno   | integer                |           | not null |
 username | character varying(100) |           | not null |
 wpm      | double precision       |           |          |
 error    | integer                |           |          |
Indexes:
    "typingtestuser_pkey" PRIMARY KEY, btree (username, testno)
Foreign-key constraints:
    "typingtestuser_username_fkey" FOREIGN KEY (username) REFERENCES users(username) ON UPDATE SET NULL ON DELETE CASCADE
```

Generated Paragraph -

```
LHhvF wrHM mdejGm MuJAD G,rH Kcohc FsVdoA fcc
H fJrDVe lv,ecii
```

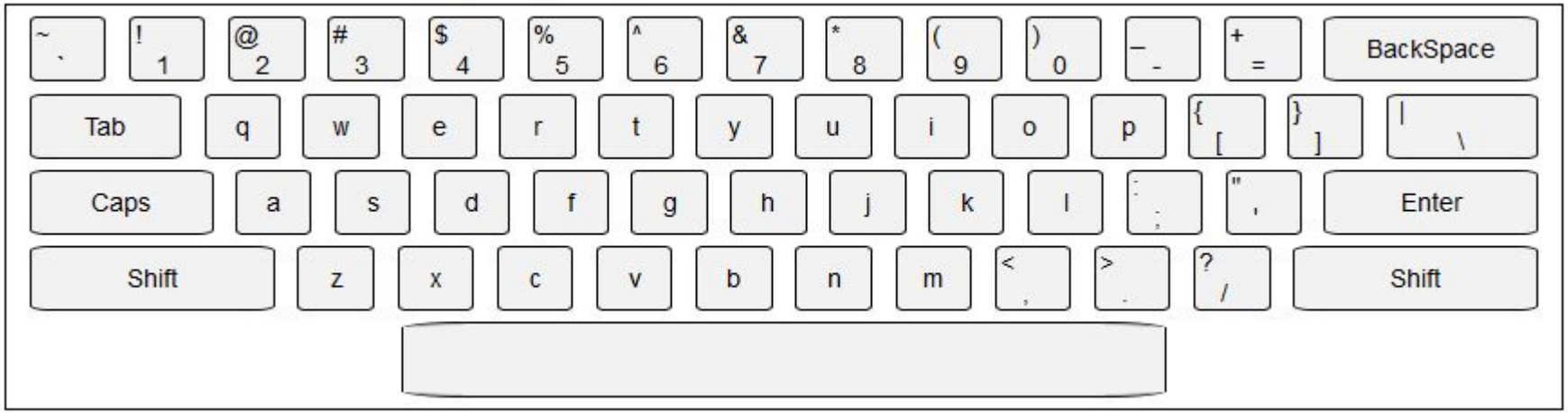
Relative Statistics-

Speed	Erorrs
21.18	1
(-18.98)	(-2.20)

Speed of the latest completed test and the Gain(Average WPM - Last WPM).

Number of errors in the latest completed test and the Gain(Average Error - Last Error).

Keyboard for visual feedback -



The respective keys light up on key press. Green if right letter is pressed, red otherwise.

Only one PHP script(testAJAX.php) handles all the requests. It returns an JSON encoded associative array - [lessonCompleted, averageSpeed, averageError, totalSamples, lastWPM, lastError]. And inserts into **`typingtestuser`** and updates **`userstats`** on completion of a test.

1. Fetch last completed lessonid

   ```
   SELECT lessonid FROM lessonsCompleted WHERE username='{$userName}'
   ```

   Last lesson is accessed as `obj["lessonCompleted"]` where `obj` is the parsed object of the returned JSON string.

2. Fetch averageSpeed, averageError and totalSamples

   ```
   SELECT * FROM userstats WHERE username='{$userName}'
   ```

   Accessed as `obj["averageSpeed"]`, `obj["averageError"]` and

```
 username | topspeed | averagespeed | averageerror | totalsamples
----------+----------+--------------+--------------+--------------
 userss   |    43.88 |        38.43 |            3 |           11
(1 row)
```

3. Fetch lastWPM and lastError

```
SELECT wpm,error FROM typingtestuser WHERE testno = (SELECT
max(testno) FROM typingtestuser WHERE username='{$userName}')
```

This query gives the wpm and error stats of latest completed test. Accessed as `obj["lastWPM"]` and `obj["lastError"]`

```
  wpm   | error
--------+-------
 21.18  |     1
(1 row)
```

4. Insert new test data into the table

```
INSERT INTO typingTestUser values({$obj["testNo"]}, '{$userName}',
{$obj["wpm"]}, {$obj["errors"]})
```

Where, `obj["testNo"]` is totalSamples+1, `obj["wpm"]` and `obj["errors"]` are the wpm and errors of the to be inserted test.

5. Insert into/Update **`userstats`** table

```
SELECT *from typingtestuser where username = '{$userName}'
```

```
SELECT count(*) FROM userstats WHERE username='{$userName}'
```

After insertion all the data is fetched and calculated to form the fields of **`userstats`** table.

```
UPDATE userstats set totalSamples={$c}, averagespeed={$speedSum},
averageerror={$errorSum}, topspeed={$topSpeed}  where username =
'{$userName}'
```

Or if it's user's first test,

```
INSERT INTO userstats VALUES('{$userName}', {$topSpeed}, {$speedSum},
{$errorSum}, {$c})
```

Typing Challenges

User can practice on famous paragraphs stored in the database to get a more real life typing practice. **`typingchallenges`** table stores all the paragraphs used. XHR requests are used to fetch the paragraphs.

Description of **`typingchallenges`** -

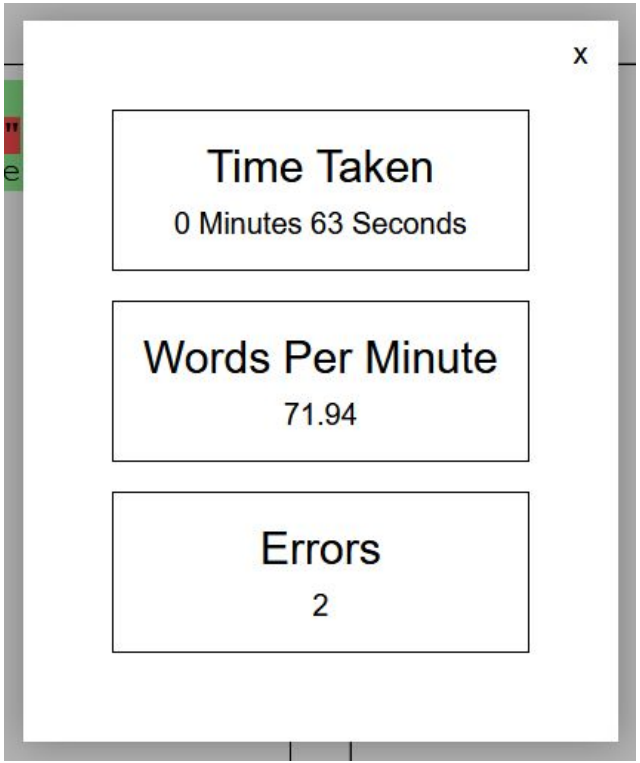
```
     Column       |       Type       | Collation | Nullable | Default
------------------+------------------+-----------+----------+---------
 typingchallengeid | double precision |           | not null |
 challengetitle    | text             |           |          |
 para              | text             |           |          |
Indexes:
    "typingchallenges_pkey" PRIMARY KEY, btree (typingchallengeid)
```

1. Fetch paragraphs

   ```
   SELECT * FROM typingchallenges
   ```

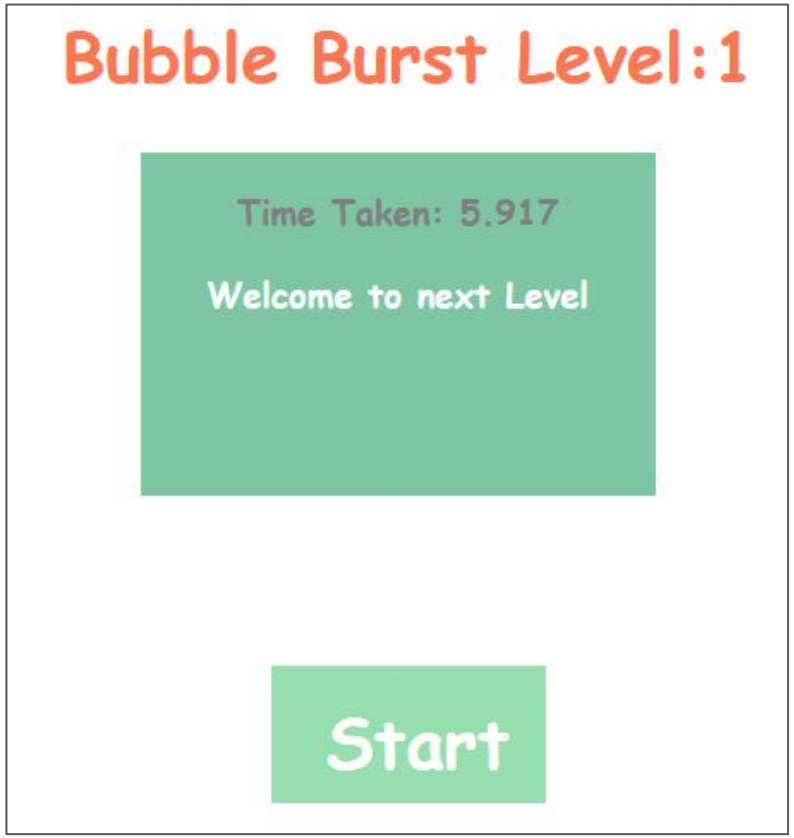
   Returns JSON encoded associative array of form [“challengeName”:”challengePara”].

Upon completing a challenge a modal with statistics is shown -



Games

Games generates scores on completion -



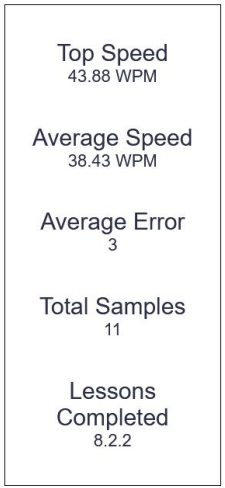
Statistics

This page summarizes and visualizes the user statistics. Tables used are **`lessonscompleted`**, **`userstats`** and **`typingtestuser`**.

1. Fetch user statistics

```
SELECT * FROM userstats WHERE username='{$userName}'
```

```
SELECT lessonid FROM lessonsCompleted WHERE username='{$userName}'
```



Results from above queries is used to construct this table present on the right hand side of the page.

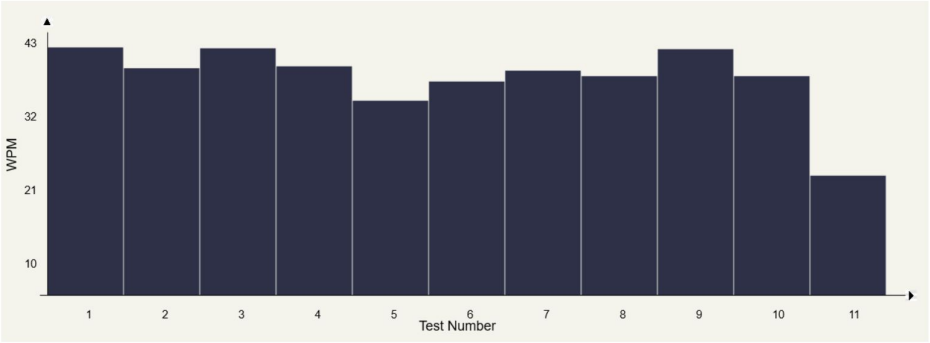
2. Fetch data for the graphs

```
SELECT * FROM typingtestuser WHERE username='{$userName}'
```

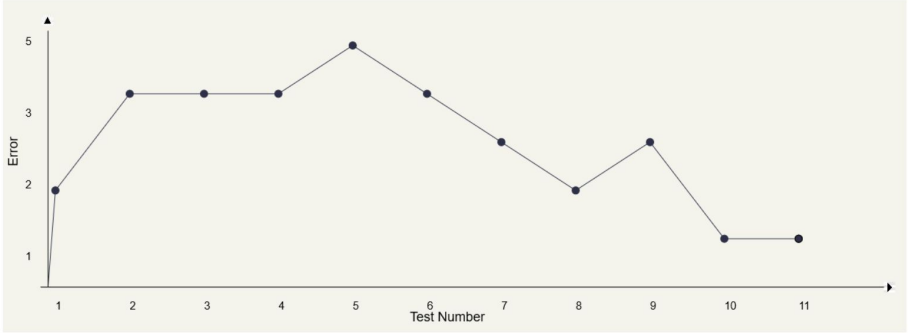
```
SELECT max(wpm) FROM typingtestuser WHERE username='{$userName}'
```

```
SELECT max(error) FROM typingtestuser WHERE username='{$userName}'
```

Two graphs “Test Number vs WPM”(histogram) and “Test Number vs Error”(line chart) are made from the results of these queries.



Test Number vs WPM



Test Number vs Error

4. Coding Style Required

Hardware Specification

1. Since this is a web application hardware doesn’t matter much, any computer that is capable of running a browser will be technically able to run this website.
2. Nor the website is demanding in graphics or computation so 2GB of RAM will be more than sufficient.
3. Neither does it require any storage on the client machine. All the data is stored on the database on the server.
4. Most of the hardware requirements are dictated by the browser not the website itself.

Platform

Operating System - Cent OS

Language - PHP, Javascript, HTML and CSS

Database - PostgreSQL

Browser - Firefox

Coding Style Required

Folder Structure-

```
TypingWizard
|-- Database/
|-- Images/
|-- Js/
|-- Styles
|-- login.php
...
```

Root of the project contains all the HTML and PHP files of the pages.

`Database` folder contains config.php, SQL for database creation, SQL for database insertion and all the PHP scripts used in AJAX.

`Js` folder contains all the javascripts files. Similarly `Styles` contains all the CSS files.

`config.php` file defines the Config class which constructs the database connection string. Static function getConnnectionString can be used to get the connection string.

HTML

Almost all the HTML is written inside PHP files for dynamic behaviour. Depending on the user action or database query result we can generate pages on server and serve them to the user. Each HTML page has its own associated CSS file.

```
<?php  if($USEREXISTS==1): ?>
    <span class="wrong">User already exists !</span>
<?php   else : ?>
    <span class="wrong">Invalid Username!</span>
<?php endif ?>
```

Using PHP to write HTML also allows us to create templates. Here in our project we include `header.html` and `footer.html` to make templates for different pages. These PHP files are written in imperative paradigm style.

CSS

1. Most of the styling is done using class names and id with occasional use of tag names directly.
2. Flexbox is used for layout.
3. Acme and system/default fonts are used.
4. Custom CSS reset is used for easier styling -

```css
* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}
```

Javascript

1. Javascript written for this project is imperative and event driven.
2. Each page has its own separate associated js file.
3. Callbacks are used for asynchronous behaviour.
4. XHR is used to fetch data from the database.
5. Various events listeners are used. (ex. ‘keypress’, ‘click’, etc).
6. Regex functions like test and match are for form validation and other tasks.
7. Large scripts are broken into functions for easier management and understanding.

Predefined interface/method

PHP

1. PostgreSQL functions -

   pg_connect, pg_query, pg_num_rows, pg_fetch_assoc, pg_fetch_row

2. JSON

   json_encode, json_decode

3. Other

   isset, header, require, explode, echo, file_get_contents, round, die

Javascript

1. JSON

   JSON.parse, JSON.stringify

2. AJAX (XMLHttpRequest Object)

   open, setRequestHeader, send, onload, responseText

3. Document

   querySelector, addEventListener, querySelectorAll, createDocumentFragment, createElement, etc.

4. Other

   toFixed, classList.add, classList.remove, appendChild, toLowerCase, etc.

5. Limitations of the System

1. Hashed Passwords

   Currently we store passwords as plain text in the database. Hashing passwords will provide better security in case of database leak and is a good practice in general.

2. Better Authentication

   Presently a cookie with value of ‘username’ is stored on the client machine. A server generated session-id will be a better suit for authentication as it will map the user to a particular login and remove the problem of multiple login.

3. Lesson Instructions

   The instructions at the beginning of each lesson would have helped user in easy understanding and completion, The user is not guided about which finger to be used to press a specific key and that can be troublesome and confusing.

4. Keyboard Layout

   The project is made keeping QWERTY keyboard in the mind though keyboards with different layouts are also available in the market.

5. Competence Feature

   Multiple users can give tests, however we don’t provide a performance metric which will compare one user’s performance with multiple users who gave the same test.

6. Multiple Languages Not Supported

   Currently ‘Typing Wizard’ is available in English only, however touch typing is not limited to English other languages can also be included.

7. Responsive Design

   The website was made with only Firefox desktop application in mind. Its UI doesn’t change and respond to other browsers on other platforms.

6. References

- W3Schools Online Web Tutorials

  Used for quick reference of topics of HTML, CSS and Javascript.

- Touch Typing Practice Online (typingstudy.com) and Typing Practice (keybr.com)

  Used as a inspiration for the design and structure of project.

- Stack Overflow

  Used to troubleshoot unexpected situations encountered during the project production.

- MDN Web Docs

  Used for in depth Javascript and CSS reference.

- diagrams.net

  Used to make flowcharts and various other diagrams for the documentation.

- Pixabay

  Used for background images present on the signup screen.

- FlatIcon

  Used for website icon.

- Coolors

  Used to generate custom color palette for website.

- PHP Docs

  Used for PHP reference.

- PostgreSQL: Documentation

  Used for PostgreSQL reference.

- Google Fonts

  Used fonts are downloaded from.