EASY EXAM

SARTHAK DABHI
006118706@coyote.csusb.edu

Follow this and additional works at: https://scholarworks.lib.csusb.edu/etd

Part of the Educational Assessment, Evaluation, and Research Commons, and the Engineering Education Commons

Recommended Citation
DABHI, SARTHAK, "EASY EXAM" (2019). Electronic Theses, Projects, and Dissertations. 793.
https://scholarworks.lib.csusb.edu/etd/793

This Project is brought to you for free and open access by the Office of Graduate Studies at CSUSB ScholarWorks. It has been accepted for inclusion in Electronic Theses, Projects, and Dissertations by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.
EASY EXAM

A Project
Presented to the
Faculty of
California State University,
San Bernardino

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Sarthak Hasmukhbhai Dabhi
March 2019
EASY EXAM

A Project
Presented to the
Faculty of
California State University,
San Bernardino

by
Sarthak Hasmukhbhai Dabhi
March 2019
Approved by:

Dr. Kerstin Voigt, Advisor, Computer Science and Engineering

Dr. Ernesto Gomez, Committee Member

Dr. Owen J. Murphy, Committee Member
ABSTRACT

Easy Exam is web-based educational software which allows professors to take and students to give exams. This project focuses on making an effortless process for professors to make an exam, to grade exams for all students, and to create class statistical analysis reports about all exam-taking students. At the end of the exam, students can see their report with analytics based on topics in the exam. This web-based software will help students to identify their weaknesses and strengths. So, students can focus on their weaknesses and improve their knowledge. Furthermore, this software will help the professors to identify which students are weak in which sections in their studies. So, professors can focus on those areas and make their student understand more. This project works on multiple choices questions. However, this project has extensive unique opportunities and abilities that will be covering many different types of questions such as programming questions in the future stages of this project.
ACKNOWLEDGEMENTS

I want to express my sincere gratitude to my advisor Dr. Kerstin Voigt for assisting and leading me for completing this project. I was a student assistant for her class. During, that time, I learned many things about exams, students, and coursework which ultimately led me to work on this project.

I also want to show my sincere gratitude to committee members Dr. Ernesto Gomez and Dr. Owen Murphy for their support. They let me have the internship in RxPrep, which is directly helped me to design and develop this project.

Also, I am thankful for our Department of Computer Science at California State University, San Bernardino. Everyone really helped me a lot by answering all doubts and questions.
# TABLE OF CONTENTS

ABSTRACT .......................................................................................................................... iii

ACKNOWLEDGEMENTS .................................................................................................. iv

LIST OF FIGURES ........................................................................................................ vii

CHAPTER ONE: INTRODUCTION

Background......................................................................................................................... 1

CHAPTER TWO: SYSTEM ANALYSIS

Proposed System ............................................................................................................. 2

System Requirement Specification .................................................................................. 3

Software Requirements .................................................................................................. 4

Dependencies .................................................................................................................. 4

CHAPTER THREE: SYSTEM DESIGN

Scenarios ......................................................................................................................... 6

Data Flow Diagram ......................................................................................................... 9

Use Case Diagram .......................................................................................................... 10

Sequence Diagram ........................................................................................................ 11

Class Diagram ................................................................................................................ 12

Entity-Relationship Model ............................................................................................. 13

Module Diagram ............................................................................................................. 14

Calculations ..................................................................................................................... 14

Mean ............................................................................................................................... 14
Median ........................................................................................................... 14
Standard Deviation ......................................................................................... 15
Exam Statistics ................................................................................................. 15
Chapter Breakdown Report .............................................................................. 15
Student Score Distribution Report ................................................................. 16

CHAPTER FOUR: SYSTEM TESTING
Faker Script ...................................................................................................... 17

CHAPTER FIVE: FEATURE ENHANCEMENT
Programmatical Question ................................................................................. 24
Question Difficulty Level .................................................................................. 24
Interactive Questions ......................................................................................... 24

CHAPTER SIX: CONCLUSION........................................................................ 25

APPENDIX A: APPLICATION CODE ............................................................. 26
APPENDIX B: PROJECT SCREENSHOTS ....................................................... 64
REFERENCES .................................................................................................. 77
LIST OF FIGURES

Figure 1. Data Flow Diagram........................................................................................................9
Figure 2. Use Case Diagram........................................................................................................10
Figure 3. Sequence Diagram.......................................................................................................11
Figure 4. Class Diagram...............................................................................................................12
Figure 5. ER Diagram................................................................................................................13
Figure 6. Module Diagram .........................................................................................................14
Figure 7. Chapter Breakdown Report.......................................................................................16
Figure 8. Student Score Normal Distribution Report ...........................................................16
Figure 9. Faker Script ..............................................................................................................23
CHAPTER ONE

INTRODUCTION

Background

Easy Exam is an educational project which helps students and professors for their exams. This project is a web-based software with a centralized database. I choose to build this project because of my experience with exams was extremely difficult. After my exams, it was a tough time to find which questions are from which chapters and to remember hard questions for final exams. Also, for professors, it is quite hard to analyze the whole class's students score and analytics. This software gives graphical and numerical reports to understand the exam and student's performance. I built this project with the Laravel PHP framework and MySQL database. I choose the Laravel PHP framework because it gives flexibility for rapid development. The objective of this project is to develop software which allows students and professors to take and provide exams. Also, this project will provide statistics for students' performance.
CHAPTER TWO

SYSTEM ANALYSIS

Proposed System

Basically, the process of the software is defined in three modules. 1. Administration, which allows administrators to add new subjects, projects, and build fake data to examine; 2. Professor, which will facilitate professors to create exams, check individual student's reports, and understand full class strengths and weaknesses; and 3. Students, which allows students to take exams.

First, the administrator will create professors accounts, subjects, and chapters. Then all processors will create questions. Professors can use those questions to form an exam, or they can use an existing question pool to create an exam. Once, questions are being used in our system. All questions will get "Correctness" score. So, the professor can determine whether they want to use those questions for their next exam or not. Also, after an exam, the professor can analyze three kinds of statistics. 1. Student's exam summary, 2. Chapter's breakdown for the entire class, and 3. Student score distribution with mean, median and standard deviation. For students, our software offers to take exams. After an exam, students can see their exam summary to understand their weakness and strength in their subjects. Also, they can identify their progress attempt by attempt. Furthermore, they can have difficult questions as their flashcard to remember and practice after exams.
System Requirement Specification

The administrator objectives for the Easy Exam are to:

1. Manage professors and users accounts
2. Manage subjects
3. Manage chapters
4. Feed fake data to the system

The professor objectives for the Easy Exam are to:

1. Manage questions
2. Manage exams
3. Assign exams to students
4. Exam analysis
   a. Student exam summary
   b. Students score distribution
   c. Chapter breakdown for an exam

The student objectives for the Easy Exam are to:

1. Take exams
2. Exam analysis
   a. Chapter breakdown for an exam
   b. Attempt progress report
   c. Exam summary
3. Flashcards
Software Requirements

1. Web server (Apache or Nginx)
2. PHP scripting language, version >= 7.1.3
3. Laravel PHP Framework
4. MySQL database
5. Visual Studio Code IDE
6. NPM, JavaScript package manager
7. Composer, Dependency Manager for PHP
8. OpenSSL PHP Extension
9. PDO PHP Extension
10. Mbstring PHP Extension
11. Tokenizer PHP Extension
12. XML PHP Extension
13. Ctype PHP Extension
14. JSON PHP Extension
15. BCMath PHP Extension

Dependencies

Followings are frameworks and libraries are used to develop this software.

1. Laravel PHP framework
Laravel is a free, open-source PHP the model–view–controller (MVC) architectural web framework. This framework gives a web-based application.

2. Faker
Faker is an open-source PHP library which generates fake data for a system. This fake data will allow the system to emulate system correct behavior.

3. Bootstrap CSS framework
Bootstrap is an open-source frontend component library which helps to design mobile-first projects. It also involves jQuery JavaScript library itself.

4. jQuery
jQuery is an open-source JavaScript library which manipulates HTML DOM tree, as well as handle event, CSS animation, and Ajax.

5. Laravel Mix
It is a Laravel PHP framework library that provides a fluent API for defining Webpack build common CSS and JavaScript pre-processors.

6. Axios
It is an open-source JavaScript library which allows sending HTTP requests to servers.
Before writing actual code for the software, designing the system architecture is very important. It helps to identify how data will flow through modules and components. This process will make it easier to identify all methods and how to write code.

Scenarios

**Login**

1. A user opens a web application in a browser.
2. Click on login button, and it will show a login screen.
3. User enters username and password to login.
4. Server authenticates the user and take a user to the dashboard as per their roles such as student, professor, and administrator.

**List and create professors**

1. Login as administrator.
2. Go to professor's list page.
3. Click on “New Professor” button, and it will show the form to create a professor account.

**List and create subjects**

1. Login as administrator.
2. Go to subject’s list page.
   
3. Click on “New Subject” button, and it will show the form to create a subject.

**List and assign chapters to a subject**

1. Login as administrator.

2. Go to any subject, and it will show chapters which are assigned to a subject.

3. You can create a new chapter and assign to the same subject.

**List and create exam**

1. Login as professor.

2. Go to exam’s list page.

3. Click on “New Exam” button, and it will show a form to create an exam.

**Assign students to exam**

1. Login as professor.

2. Go to an exam list page.

3. Click on “Add Students”, and it will show a list of students to add students to an exam.

**List and create a question**

1. Login as professor.

2. Go to the questions list page.

3. Click on “New Question”, and it will show a form to create a professor account.
Assign questions to exam

1. Login as professor.
2. Go to an exam list page.
3. Click on “Add Questions”, and it will show a list of questions to add questions to an exam.

Exam report

1. Login as professor.
2. Click on an exam list page.
3. Click on “Exam Statistics”, and it will show a report for the exam.

Take exam

1. Login as student
2. Click on an exam list page
3. Click on “Take Exam”, and it will show questions for the exam.

Attempt report

1. Login as student
2. Click on an exam list page
3. Click on “Attempts” to see a report.
Data Flow Diagram

This diagram will help to understand how data is flowing through processes in our software.

Figure 1. Data Flow Diagram
Use Case Diagram

This diagram will help to understand how a user will interact with our software in terms of their roles and actions.

Figure 2. Use Case Diagram
Sequence Diagram

This diagram will help to understand how a user will interact with our software and other types of user in terms of their roles and actions. Here below this sequence diagram will show how professor can create exam and assign to student. After exams to available to students, they can take it and see their summary report.

![Sequence Diagram](image)

Figure 3. Sequence Diagram
Class Diagram

This diagram will help to understand structure of software using class’s instances and methods. From class diagram, we can identify how this MVC framework will map with its routes.

Figure 4. Class Diagram
Entity-Relationship Model

This model will help to understand database schema is working with its tables and relationships.

Figure 5. ER Diagram
Module Diagram

This diagram will help to understand at abstract level how and which modules interact.

![Module Diagram](image)

Figure 6. Module Diagram

Calculations

This software uses a few mathematical formulas to derive report and exam statistics. For each exam, we derive mean, median, and standard deviation.

**Mean**

Mean is the average of the numbers.

e.g.

List of numbers = 1, 2, 3, 4, 5

Mean = \( (1 + 2 + 3 + 4 + 5) / 5 = 3 \)

**Median**

Median is the middle number of ordered numbers.
e.g.
List of numbers = 1, 2, 3, 4, 5
Median = 3

**Standard Deviation**
Standard deviation is a measure of how spread numbers are.
e.g.
List of numbers = 1, 2, 3, 4, 5
Mean = 3
Standard deviation = \( \sqrt{\frac{((3-1)(3-1) + (3-2)(3-2) + (3-3)(3-3) + (3-4)(3-4) + (3-5)(3-5))/5} } = 1.4142 \)

**Exam Statistics**
Easy Exam provides three kinds of reports chapter breakdown report, students score distribution report, and student’s exam summary report.

**Chapter breakdown report**
This report will chapter wise score report for entire class. It derives from students score for particular question. It will calculate chapter score from all questions in that exam. Here is live exam chapter breakdown report.
Student score distribution report

To derive this report, our software calculates mean and standard deviation for entire class students for an exam. These two values put in chart and it will give us standard normal distribution chart.

Figure 7. Chapter Breakdown Report

Figure 8. Student Score Normal Distribution Report
CHAPTER FOUR
SYSTEM TESTING

For this software, testing is done in every phase of the development. This web-based application needs manual testing. This software requires a right amount of data to test. I have made a data feeder script which put a huge amount of data into a database. So, we can do manual testing.

Faker Script

class FakerController extends Controller
{
    private $request, $faker;

    public function __construct(Request $request, Faker $faker)
    {
        $this->request = $request;
        $this->faker = $faker;
    }

    public function dummyData()
    {
        $this->fakeAdmins(1);
        $this->fakeProfessors(3);
        $this->fakeStudents(50);
        $this->fakeSubjects(5);
        $this->fakeQuestions(50);
        $this->fakeExams(15);
        $this->fakeAllowUserExams(50);
        $this->fakeTakeExam(5, 45);
    }
}
public function fakeAdmins($number = 1)
{
    for($i = 0; $i < $number; $i++)
    {
        User::create(
            ['name' => $this->faker->name,
             'email' => $this->faker->userName . '@csusb.edu',
             'password' => Hash::make('password'),
             'role' => 8
        ]);
    }
}

public function fakeProfessors($number = 1)
{
    for($i = 0; $i < $number; $i++)
    {
        User::create(
            ['name' => $this->faker->name,
             'email' => $this->faker->userName . '@csusb.edu',
             'password' => Hash::make('password'),
             'role' => 1
        ]);
    }
}

public function fakeStudents($number = 1)
{
    for($i = 0; $i < $number; $i++)
    {
        User::create(
            ['name' => $this->faker->name,
             'email' => $this->faker->userName . '@csusb.edu',
             'password' => Hash::make('password'),
             'role' => 0
        ]);
    }
}
public function fakeSubjects($number = 1)
{
    for($i = 0; $i < $number; $i++)
    {
        $subject = Subject::create([
            'number' => $this->faker->numerify('CSE ###'),
            'name' => $this->faker->word . ' ' . $this->faker->word,
        ]);[
        $this->fakeChapters(10, $subject->id);
    }
}

public function fakeChapters($number = 1, $subject_id)
{
    for($i = 0; $i < $number; $i++)
    {
        Chapter::create([[
            'subject_id' => $subject_id,
            'chapter_number' => ($i + 1),
            'name' => $this->faker->word . ' ' . $this->faker->word,
        ]]);[
    }
}

public function fakeQuestions($number = 1)
{
    for($i = 0; $i < $number; $i++)
    {
        $question = Question::create([[
            'question' => $this->faker->paragraph(rand(3,5)),
            'explanation' => $this->faker->paragraph(rand(3,5)),
            'created_by' => User::where('role', 1)->first()->id,
        ]]);[
        $subject = Subject::first()->id;
    }
$chapters = Chapter::where('subject_id', $subject)->get();
$temp = array();

foreach($chapters as $chapter)
{
    array_push($temp, $chapter->id);
}
$questionDetail = QuestionDetail::create([
    'question_id' => $question->id,
    'subject_id' => Subject::first()->id,
    'chapter_id' => $temp[rand(1, count($temp) - 1)],
]);
for($j = 0; $j < 4; $j++)
{
    $questionAnswer = QuestionAnswer::create([
        'question_id' => $question->id,
        'answer' => $this->faker->paragraph(1),
        'is_correct' => ($j == 0) ? 1 : 0,
    ]);
}
}

public function fakeExams($number = 1)
{
    for($i = 0; $i < $number; $i++)
    {
        $allow_retake = $i % 2;
        $exam = Exam::create([
            'name' => $this->faker->sentence(2),
            'description' => $this->faker->paragraph(1),
            'status' => 1,
            'show_score' => 1,
            'allow_retake' => $allow_retake,
            'show_summary' => 1,
            'show_explanation' => (!$allow_retake,
            'start_datetime' => null,
        );
    }
}
'end_datetime' => null,
]);

for($j = 1; $j < rand(45, 50); $j++)
{
    $examQuestion = ExamQuestion::create(
        'exam_id' => $exam->id,
        'question_id' => $j,
    );
}
}

public function fakeAllowUserExams($number = 1)
{
    $exams = Exam::get();
    foreach($exams as $exam)
    {
        $users = User::where('role', 0)->get();
        foreach($users as $user) {
            AllowUserExam::create(
                'exam_id' => $exam->id,
                'user_id' => $user->id
            );
        }
    }
}

public function fakeTakeExam($id, $number = 1)
{
    $user = User::find($id);
    $allowUserExams = $user->allowUserExams;
    foreach($allowUserExams as $keyy => $allowUserExam)
    {
        $exam = Exam::find($allowUserExam->exam_id);
        if($exam->allow_retake == 1)
            $attempts = rand(1, 5);
        else
            $attempts = 1;
    }
for($i = 1; $i < $attempts; $i++)
{
    $userExam = UserExam::create([  
        'user_id' => $user->id,  
        'exam_id' => $exam->id,  
        'attempt' => $i,  
        'status' => 0
    ]);  
    $answers = array();  
    foreach($exam->examQuestions as $examQuestion)  
    {
        foreach($examQuestion->question->answers as $answer)
        {
            array_push($answers, $answer->id);
        }
        $answer_id = $answers[rand(0, 3)];  
        $userExamAnswer = UserExamAnswer::create([  
            'user_exam_id' => $userExam->id,  
            'question_id' => $examQuestion->question_id,  
            'question_answer_id' => $answer_id,  
            'is_correct' => QuestionAnswer::find($answer_id)->is_correct,
        ]);  
    }
    $userExamAnswers = UserExamAnswer::where('user_exam_id', $userExam->id)->get();  
    $total = 0;  
    foreach($userExamAnswers as $userExamAnswer) {
        $total += $userExamAnswer->is_correct;
    }
    $userExam->update(['status' => 1, 'score' => $total]);
}
$id++;
if($id < $number)
    $this->fakeTakeExam($id, $number);
else
    return "DONE!";
CHAPTER FIVE

FEATURE ENHANCEMENT

There are numerous features that can be implemented to enhance our application from which few are listed below:

Programmatical Question

For this feature, we can add new questions such as programmatically questions where students can write the part of the program and check the output after compiling on a server.

Question Difficulty Level

For this feature, we can implement with the current exam questions history. We can identify the question’s strength, and we can add or remove in ongoing exam.

Interactive Questions

For this feature, we can implement with HTML5 game elements which will make all questions interactive. This feature also helps to explain the questions and answer with understanding.
CHAPTER SIX

CONCLUSION

Easy Exam is educational software for professors and students. This software will save time and efforts for professors and students to understand the exam and student's behavior for exams. This project will give detailed analytics and report for exams. This software will help students to understand where they can focus to gain more score in exams. Also, professors will understand how students did for their exams. This project has the potential of many great new features such as interactive questions, programmatical questions, and question difficulty level. This project will be also useful for student's assessment for any entry-level exams.
APPENDIX A

APPLICATION CODE
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Http\Requests\ProfessorRequest;
use App\Http\Requests\SubjectRequest;
use App\Http\Requests\DeleteProfessorRequest;
use App\Http\Requests\UpdateProfessorRequest;
use App\Http\Requests\UpdateProfessorPasswordRequest;
use App\Http\Requests\DeleteSubjectRequest;
use App\Http\Requests\UpdateSubjectRequest;
use App\Http\Requests\ChapterRequest;
use App\Http\Requests\UpdateChapterRequest;
use App\Subject;
use App\Chapter;
use App\User;
use App\Exam;

class AdminController extends Controller
{
    private $request;

    public function __construct(Request $request)
    {
        $this->request = $request;
    }

    public function updateChapter(UpdateChapterRequest $request)
    {
        $request->validated();

        $chapter = Chapter::find($request->id);
        if($chapter->update($request->all()))
            return redirect()->route('admin.edit-chapter', ['id' => $chapter->id]);
    }
}
public function editChapter($id)
{
    $chapter = Chapter::find($id);
    return view('admin.pages.edit-chapter', ['chapter' => $chapter]);
}

public function deleteChapter()
{
    $chapter = Chapter::find($this->request->id);
    if($chapter->questions->count() == 0)
    {
        if($chapter->delete())
        {
            return redirect()->route('admin.chapters', ['id' => $chapter->subject_id]);
        }
    }
}

public function addChapter(ChapterRequest $request)
{
    $request->validated();
    if(Chapter::create(
        ['subject_id' => $request->subject_id,
         'chapter_number' => $request->chapter_number,
         'name' => $request->name,
         ]
    ))
    {
        return redirect()->route('admin.chapters', ['id' => $request->subject_id]);
    }
}

public function chapters($id)
{
    $subject = Subject::find($id);
return view('admin.pages.chapters', ['subject' => $subject]);
}

public function home()
{
    $professors = User::where('role', 1)->get()->count();
    $students = User::where('role', 0)->get()->count();
    $subjects = Subject::get()->count();
    $exams = Exam::get()->count();
    return view('admin.pages.home', ['professors' => $professors, 'students' => $students, 'subjects' => $subjects, 'exams' => $exams]);
}

public function subjects()
{
    $subjects = Subject::latest()->get();
    return view('admin.pages.subjects', ['subjects' => $subjects]);
}

public function professors()
{
    $professors = User::where('role', 1)->latest()->get();
    return view('admin.pages.professors', ['professors' => $professors]);
}

public function newProfessor()
{
    return view('admin.pages.new-professor');
}

public function newSubject()
{
    return view('admin.pages.new-subject');
}
public function createProfessor(ProfessorRequest $request)
{
    $request->validated();

    if(User::create(['
        'name' => $request->name,
        'email' => $request->email,
        'password' => bcrypt($request->password),
        'role' => 1,
    ]))
    {
        return redirect()->route('admin.professors');
    }
}

public function createSubject(SubjectRequest $request)
{
    $request->validated();

    if(Subject::create(['
        'number' => $request->number,
        'name' => $request->name,
    )))
    {
        return redirect()->route('admin.subjects');
    }
}

public function editSubject($id)
{
    $subject = Subject::where('id', $id)->first();
    return view('admin.pages.edit-subject', ['subject' => $subject]);
}

public function editProfessor($id)
{
$professor = User::where(['id' => $id, 'role' => 1])->first();
    return view('admin.pages.edit-professor', ['professor' => $professor]);
}

public function updateProfessor(UpdateProfessorRequest $request) {
    $request->validated();
    if (User::where('id', $request->id)->first()->update($request->all()))
        return redirect()->route('admin.professors');
}

public function updateSubject(UpdateSubjectRequest $request) {
    $request->validated();
    if (Subject::where('id', $request->id)->first()->update($request->all()))
        return redirect()->route('admin.subjects');
}

public function deleteSubject(DeleteSubjectRequest $request) {
    $request->validated();
    if (Subject::where('id', $request->id)->first()->delete())
        return redirect()->route('admin.subjects');
}

public function deleteProfessor(DeleteProfessorRequest $request) {
    $request->validated();
    if (User::where('id', $request->id)->first()->delete())
        return redirect()->route('admin.professors');
```php
public function updateProfessorPassword(UpdateProfessorPasswordRequest $request)
{
    $request->validated();

    if (User::where('id', $request->id)->first()->update(['password' => bcrypt($request->password)]))
    {
        return redirect()->route('admin.professors');
    }
}
```

Figure 11. AdminController.php

ProfessorController.php

```php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Question;
use App\QuestionDetail;
use App\QuestionAnswer;
use App\Exam;
use App\Subject;
use App\ExamQuestion;
use App\User;
use App\UserExam;
use App\Http\Requests\CreateQuestionRequest;
use App\Http\Requests\CreateExamRequest;
```
class ProfessorController extends Controller
{
    private $request;

    public function __construct(Request $request)
    {
        $this->request = $request;
    }

    public function summary($user_exam_id)
    {
        $userExam = UserExam::find($user_exam_id);
        $exam = $userExam->exam;
        $score = $userExam->score;
        $examQuestions = $exam->examQuestions;

        $tempArray = array();
        foreach($examQuestions as $examQuestion)
        {
            $temp = array();
            $temp['question_text'] = $examQuestion->question;
            $temp['explanation'] = $examQuestion->question->explanation;
            $is_correct = UserExamAnswer::where(['user_exam_id' => $userExam->id, 'question_id' => $examQuestion->question_id])->first();
            $temp['is_correct'] = ($is_correct === NULL) ? NULL : $is_correct->is_correct;
            $temp['chapter'] = $examQuestion->question->questionDetails->chapter->name;
            $tempArray = array_prepend($tempArray, $temp);
        }
    }
}
$show_questions = $tempArray;

$tempCollection = collect($show_questions);
$groupedChapterQuestions = $tempCollection->groupBy('chapter')->toArray();

$chapterData = array();
foreach($groupedChapterQuestions as $chapter => $chapterQuestions)
{
    $correct = 0;
    foreach($chapterQuestions as $chapterQuestion)
    {
        if ($chapterQuestion['is_correct'] == 1)
            $correct++;
    }
    $percentage = ($correct / count($chapterQuestions)) * 100;
    $chapterData = array_prepend($chapterData, ['chapter' => $chapter, 'total_questions' => count($chapterQuestions), 'percentage' => $percentage]);
}

return view('professor.pages.summary', ['total_percentage' => ($score / count($examQuestions)) * 100, 'show_questions' => $show_questions, 'exam' => $exam, 'chapter_data' => $chapterData]);

public function statistics($id)
{
    $exam = Exam::find($id);
    $allowUserExams = AllowUserExam::where('exam_id', $id)->get();
    $userExams = UserExam::where('exam_id', $id)->get();
    $userExamsScores = array();
    $totalStudents = 0;
    $mean = 0;
    $median = 0;
$sd = 0;
$totalScore = 0;
$totalStudents = $userExams->groupBy('user_id')- >count();

$tempQuestions = array();
foreach($exam->examQuestions as $examQuestion)
{
    $userExams = UserExam::where('exam_id', $exam->id)- >get();
    $tempQuestions[$examQuestion->question_id]['correct'] = 0;
    $tempQuestions[$examQuestion->question_id]['chapter_id'] = $examQuestion->question->questionDetails->chapter->id;
    $tempQuestions[$examQuestion->question_id]['total'] = 0;
    foreach($userExams as $userExam)
    {
        $userExamAnswers = UserExamAnswer::where([ 'user_exam_id' => $userExam->id, 'question_id' => $examQuestion->question->id])- >get();
        foreach($userExamAnswers as $userExamAnswer)
        {
            $tempQuestions[$userExamAnswer->question_id]['correct'] += $userExamAnswer->is_correct;
        }
        $tempQuestions[$userExamAnswer->question_id]['total'] += count($userExamAnswers);
    }
    $tempQuestions[$examQuestion->question_id]['score'] = 0;
    if($tempQuestions[$examQuestion->question_id]['total'] > 0)
    {
        $tempQuestions[$examQuestion->question_id]['score'] = number_format((($tempQuestions[$examQuestion->question_id]['correct'] / $tempQuestions[$examQuestion->question_id]['total']) * 100, 2);
    }
    $groupedChapterData = collect($tempQuestions)- >groupBy('chapter_id')->toArray();
    $chapterScore = array();
}
foreach($groupedChapterData as $key => $groupedChapter) {
    $total = 0;
    foreach($groupedChapter as $question) {
        $total += $question['score'];
    }
    if(count($groupedChapter) > 0) {
        $chapterScore[][ucwords(Chapter::find($key)->name)] = (float) number_format(($total / count($groupedChapter)), 2);  
    }
} 

foreach($userExams as $userExam) {
    $totalScore += (($userExam->score / count($exam->examQuestions)) * 100);
    array_push($userExamsScores, ($userExam->score / count($exam->examQuestions)) * 100);
}

if(count($userExams) > 0) {
    $mean = $totalScore / count($userExams);

    $sortedScores = $userExamsScores;
    sort($sortedScores);
    $start = 0;
    $end = 0;
    $total = count($sortedScores);
    if($total > 0) {
        if(($total % 2) === 0) {
            $first = ($total/2) - 1;
            $end = ($total/2);
            $median = ($sortedScores[$first] + $sortedScores[$end])/2;
} else {
    $median = $sortedScores[($total/2) - 1];
}

$tempSD = 0;
foreach($userExamsScores as $userExamsScore) {
    $tempSD += (($mean - $userExamsScore) * ($mean - $userExamsScore));
}

if($total > 0)
    $sd = $tempSD / $total;

return view('professor.pages.statistics', [ 'exam' => $exam, 'allowUserExams' => $allowUserExams, 'userExams' => $userExams, 'mean' => $mean, 'median' => $median, 'sd' => sqrt($sd), 'userExamsScores' => $userExamsScores, 'totalStudents' => $totalStudents, 'chapterScore' => $chapterScore ]);}

public function removeStudentExam() {
    $allowUserExam = AllowUserExam::where([ 'exam_id' => $this->request->exam_id, 'user_id' => $this->request->user_id, ])->first();
    if($allowUserExam->delete()) {
    }
return json_encode($allowUserExam);
}
}

public function addStudentExam()
{
    $allowUserExam = AllowUserExam::create([%
        'exam_id' => $this->request->exam_id,
        'user_id' => $this->request->user_id,
    ]);%
    if($allowUserExam)
    {
        return json_encode($allowUserExam);
    }
}

public function examStudents($id)
{
    $examUsers = Exam::find($id)->examUsers;
    $result = array();
    foreach($examUsers as $examUser)
    {
        $user = $examUser->user;
        $temp = [
            'id' => $user->id,
            'name' => $user->name,
        ];
        $result = array_prepend($result, $temp);
    }
    return json_encode($result);
}

public function totalStudents($id)
{
    $examUsers = Exam::find($id)->examUsers;
    $userIds = array();
    foreach($examUsers as $examUser)
    {
array_push($userIds, $examUser->user_id);
}
$users = User::where('role', 0)->get();
$result = array();
foreach($users as $user)
{
    if(in_array($user->id, $userIds))
        continue;

    $temp = [
        'id' => $user->id,
        'name' => $user->name,
    ];
    $result = array_prepend($result, $temp);
}
return json_encode($result);

public function removeQuestionExam()
{
    $examQuestion = ExamQuestion::where(["exam_id" => $this->request->exam_id, 'question_id' => $this->request->question_id, ])->first();
    if($examQuestion->delete())
    {
        return json_encode($examQuestion);
    }
}

public function addQuestionExam()
{
    $examQuestion = ExamQuestion::create(["exam_id" => $this->request->exam_id, 'question_id' => $this->request->question->question_id, ]);,
    if($examQuestion)
    {

public function totalQuestionsJSON($id)
{
    $examQuestions = Exam::find($id)->examQuestions;
    $questionIds = array();
    foreach($examQuestions as $examQuestion)
    {
        array_push($questionIds, $examQuestion->question_id);
    }
    $questions = Question::get();
    $result = array();
    foreach($questions as $question)
    {
        if(in_array($question->id, $questionIds))
            continue;

        $temp = [
            'id' => $question->id,
            'question' => $question->question,
            'subject' => (!empty($question->questionDetails)) ?
                $question->questionDetails->subject->name : null,
            'chapter' => (!empty($question->questionDetails)) ?
                $question->questionDetails->chapter->name : null,
            'created_by' => $question->createdBy->name,
            'correctness' => $question->correctness()
        ];
        $result = array_prepend($result, $temp);
    }
    return json_encode($result);
}

public function examQuestionsJSON($id)
{
    $examQuestions = Exam::find($id)->examQuestions;
    $result = array();
foreach($examQuestions as $examQuestion)
{
    $question = $examQuestion->question;
    $temp = [
        'id' => $question->id,
        'question' => $question->question,
        'subject' => (!empty($question->questionDetails)) ? $question->questionDetails->subject->name : null,
        'chapter' => (!empty($question->questionDetails)) ? $question->questionDetails->chapter->name : null,
        'created_by' => $question->createdBy->name,
        'correctness' => $question->correctness()
    ];
    $result = array_prepend($result, $temp);
}
return json_encode($result);

public function addStudents($id)
{
    $exam = Exam::find($id);
    return view('professor.pages.exam-users', ['exam' => $exam]);
}

public function addQuestions($id)
{
    $exam = Exam::find($id);
    return view('professor.pages.add-questions', ['exam' => $exam]);
}

public function deleteExam()
{
    $exam = Exam::find($this->request->id);
    if($exam->status == 0)
    {
        if($exam->delete())
        {
            // Process deletion
        }
    }
}
{ return redirect()-&gt;route('professor.exams'); }}

public function updateExam(CreateExamRequest $request)
{
    $request-&gt;validated();

    $exam = Exam::find($request-&gt;id);
    $examUpdated = $exam-&gt;update([ 'name' =&gt; $request-&gt;name,
        'description' =&gt; $request-&gt;description,
        'status' =&gt; $request-&gt;status,
        'show_score' =&gt; (empty($request-&gt;show_score)) ? 0 : 1,
        'allow_retake' =&gt; (empty($request-&gt;allow_retake)) ? 0 : 1,
        'show_summary' =&gt; (empty($request-&gt;show_summary)) ? 0 : 1,
        'show_explanation' =&gt; (empty($request-&gt;show_explanation)) ? 0 : 1,
        'start_datetime' =&gt; (empty($request-&gt;start_datetime)) ? null : Carbon::parse($request-&gt;start_datetime),
        'end_datetime' =&gt; (empty($request-&gt;end_datetime)) ? null : Carbon::parse($request-&gt;end_datetime),
    ]);  
    if($examUpdated)
        return redirect()-&gt;route('professor.edit-exam', ['id' =&gt; $request-&gt;id]);
}

public function editExam($id)
{
    $exam = Exam::find($id);
    return view('professor.pages.edit-exam', ['exam' =&gt; $exam]);
public function exam($id)
{
    return Exam::find($id);
}

public function createExam(CreateExamRequest $request)
{
    $request->validated();

    $exam = Exam::create(
        ['name' => $request->name,
         'description' => $request->description,
         'status' => $request->status,
         'show_score' => (empty($request->show_score)) ? 0 : 1,
         'allow_retake' => (empty($request->allow_retake)) ? 0 : 1,
         'show_summary' => (empty($request->show_summary)) ? 0 : 1,
         'show_explanation' => (empty($request->show_explanation)) ? 0 : 1,
         'start_datetime' => (empty($request->start_datetime)) ? null : Carbon::parse($request->start_datetime),
         'end_datetime' => (empty($request->end_datetime)) ? null : Carbon::parse($request->end_datetime),
    ]);
    if($exam)
        return redirect()->route('professor.exams');
}

public function newExam()
{
    return view('professor.pages.new-exam');
}

public function deleteQuestion()
$question = Question::find($this->request->id);
if($question->delete())
{
    QuestionDetail::where('question_id', $question->id)->delete();
    QuestionAnswer::where('question_id', $question->id)->delete();
    
    return redirect()->route('professor.questions');
}

public function updateQuestion(CreateQuestionRequest $request)
{
    $request->validated();

    $questionUpdated = $question->update(['
    'question' => $request->question,
    'explanation' => $request->explanation,
    // 'created_by' => $request->user()->id,
    'created_by' => 1,
    ]); 

    if($questionUpdated)
    {
        if(!empty($request->subject_id))
        {
            if(empty($request->question_detail_id))
            {
                $questionDetail = QuestionDetail::create(['
                'question_id' => $question->id,
                'subject_id' => $request->subject_id,
                'chapter_id' => $request->chapter_id,  

                ]); 
            }
        }
    }
}
$questionDetail = QuestionDetail::find($request->question_detail_id);
    $questionDetail->update(
        [
            'subject_id' => $request->subject_id,
            'chapter_id' => $request->chapter_id,
        ]);
    }
    }
else
{
    if(!empty($request->question_detail_id))
    {
        $questionDetail = QuestionDetail::find($request->question_detail_id);
        $questionDetail->delete();
    }
    }

    foreach($request->answers as $key => $answer)
    {
        $questionAnswer = QuestionAnswer::find($key);
        $questionAnswer->update(
            [
                'question_id' => $question->id,
                'answer' => $answer,
                'is_correct' => ($key == $request->is_correct) ?
                true : false,
            ]);}
    }

    return redirect()->route('professor.question', ['id' => $question->id]);
}

public function editQuestion($id)
{
    $question = Question::find($id);
    $subjects = Subject::get();
return view('professor.pages.edit-question', ['question' => $question, 'subjects' => $subjects]);

public function question($id)
{
    $question = Question::find($id);
    return view('professor.pages.question', ['question' => $question]);
}

public function createQuestion(CreateQuestionRequest $request)
{
    $request->validated();

    $question = Question::create([ 'question' => $request->question, 'explanation' => $request->explanation, // 'created_by' => $request->user()->id, 'created_by' => 1, 'subject_id' => $request->subject_id, 'chapter_id' => $request->chapter_id, ]);}

if($question)
{
    if(!empty($request->subject_id))
    {
        $questionDetail = QuestionDetail::create([ 'question_id' => $question->id, 'subject_id' => $request->subject_id, 'chapter_id' => $request->chapter_id, ]);}
}

foreach($request->answers as $key => $answer) 
{
    QuestionAnswer::create([ 'question_id' => $question->id, 'answer' => $answer, ]);
'is_correct' => ($key == $request->is_correct) ?
true : false,
]);

return redirect()->route('professor.new-question');
}
}

public function newQuestion()
{
    $subjects = Subject::get();
    return view('professor.pages.new-question', ['subjects' => $subjects]);
}

public function home()
{
    $students = User::where('role', 0)->get()->count();
    $subjects = Subject::get()->count();
    $exams = Exam::get()->count();
    $questions = Question::get()->count();
    return view('professor.pages.home', ['students' => $students, 'subjects' => $subjects, 'exams' => $exams, 'questions' => $questions]);
}

public function exams()
{
    $exams = Exam::latest()->get();
    return view('professor.pages.exams', ['exams' => $exams]);
}

public function questions()
{
    $questions = Question::latest()->get();
    return view('professor.pages.questions', ['questions' => $questions]);
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Exam;
use App\UserExam;
use App\AllowUserExam;
use App\UserExamAnswer;
use Auth;
use App\User;
use App\Flashcard;

class AppController extends Controller
{
    private $request;

    public function __construct(Request $request)
    {
        $this->request = $request;
    }

    public function flashcards()
    {
        $flashcards = Flashcard::where('user_id', Auth::id())->latest()->get();
        return view('app.pages.flashcards', ['flashcards' => $flashcards]);
    }

Figure 12. ProfessorController.php
public function addFlashcard()
{
    $flashcard = Flashcard::where(['user_id' => Auth::id(),
                                   'question_id' => $this->request->question_id])|--get();

    if(count($flashcard) == 0)
    {
        $flashcard = Flashcard::create([
            'user_id' => Auth::id(),
            'question_id' => $this->request->question_id
        ]);  
        return "added";
    }
    else
    {
        $flashcard[0]->delete();
        return "removed";
    }
}

public function attempts($exam_id, $user_id)
{
    $userExams = UserExam::where(['exam_id' => $exam_id,
                                   'user_id' => $user_id, 'status' => 1])|--get();
    $mean = 0;
    $median = 0;
    $attemptArray = array();
    $scoresArray = array();
    $totalScore = 0;
    $tempSD = 0;
    $sd = 0;

    foreach($userExams as $userExam)
    {
        $totalScore += ((((userExam->score / count($userExam->exam->examQuestions))) * 100));
    }
array_push($attemptArray, "Attempt-" . $userExam->attempt);
array_push($scoresArray, (($userExam->score / count($userExam->exam->examQuestions)) * 100));

if(count($userExams) != 0)
    $mean = $totalScore / count($userExams);

$sortedScores = $scoresArray;
sort($sortedScores);

$start = 0;
$end = 0;
$total = count($sortedScores);
if($total > 0)
{
    if(($total % 2) === 0)
    {
        $first = ($total/2) - 1;
        $end = ($total/2);
        $median = ($sortedScores[$first] + $sortedScores[$end])/2;
    }
    else
    {
        $median = $sortedScores[(($total/2) - 1];
    }
}

foreach($userExams as $userExam)
{
    $tempSD += (($mean - (((($userExam->score / count($userExam->exam->examQuestions)) * 100))) * ($mean - (((($userExam->score / count($userExam->exam->examQuestions)) * 100)))))
}
if ($total > 0) {
    $sd = $tempSD / $total;
}
return view('app.pages.attempts', ['sd' => sqrt($sd), 'median' => $median, 'mean' => $mean, 'userExams' => $userExams, 'exam' => Exam::find($exam_id)]);

public function summary($user_exam_id) {
    $userExam = UserExam::find($user_exam_id);
    $exam = $userExam->exam;
    $score = $userExam->score;
    $examQuestions = $exam->examQuestions;

    $tempArray = array();
    foreach ($examQuestions as $examQuestion) {
        $temp = array();
        $temp['question_text'] = $examQuestion->question;
        $temp['explanation'] = $examQuestion->question;
        $is_correct = UserExamAnswer::where(['user_exam_id' => $userExam->id, 'question_id' => $examQuestion->question_id])->first();
        $temp['is_correct'] = ($is_correct === NULL) ? NULL : $is_correct->is_correct;
        $temp['chapter'] = $examQuestion->questionDetails->chapter->name;
        $tempArray = array_prepend($tempArray, $temp);
    }
    $show_questions = $tempArray;
    $tempCollection = collect($show_questions);
$groupedChapterQuestions = $tempCollection->groupBy('chapter')->toArray();

$chapterData = array();
foreach($groupedChapterQuestions as $chapter => $chapterQuestions) {
    $correct = 0;
    foreach($chapterQuestions as $chapterQuestion) {
        if ($chapterQuestion['is_correct'] == 1) $correct++;
    }
    $percentage = ($correct / count($chapterQuestions)) * 100;
    $chapterData = array_prepend($chapterData, ['chapter' => $chapter, 'total_questions' => count($chapterQuestions), 'percentage' => $percentage]);
}

return view('app.pages.summary', ['total_percentage' => ($score / count($examQuestions)) * 100, 'show_questions' => $show_questions, 'exam' => $exam, 'chapter_data' => $chapterData]);

public function endUserExam() {
    $userExam = UserExam::find($this->request->id);
    $userExamAnswers = UserExamAnswer::where('user_exam_id', $userExam->id)->get();
    $score = 0;
    foreach($userExamAnswers as $userExamAnswer) {
        if ($userExamAnswer->is_correct == 1) $score++;
    }

    if($userExam->update(['status' => 1, 'score' => $score]))
    return $userExam;
}

public function saveUserExamAnswer()
{
    $userExamAnswer = UserExamAnswer::create(
        'user_exam_id' => $this->request->user_exam_id,
        'question_id' => $this->request->question_id,
        'question_answer_id' => $this->request->question_answer_id,
        'is_correct' => $this->request->is_correct,
    );
    return $userExamAnswer;
}

public function takeExam($id)
{
    $exam = Exam::find($id);
    if($exam->status == 0)
        return redirect()->route('app.exams');

    $examQuestions = $exam->examQuestions;
    $questions = array();
    foreach($examQuestions as $examQuestion)
    {
        $q = $examQuestion->question;
        $answers = array();
        foreach($q->answers as $ans)
        {
            $tempAns = [
                'id' => $ans->id,
                'answer' => $ans->answer,
                'is_correct' => $ans->is_correct,
            ];
            $answers = array_prepend($answers, $tempAns);
        }
        $questions = array_push($questions, $temp


'id' => $q->id,
'question' => $q->question,
'explanation' => $q->explanation,
'answers' => $answers,
'selected_answer' => null,
'is_correct' => null,
'is_answered' => null,
'flashcard' => Flashcard::where(['user_id' => Auth::id(), 'question_id' => $q->id])-get()-count() > 0
];
$questions = array_prepend($questions, $temp);
}

$total = count($questions);
$totalAnswered = 0;

$userExams = UserExam::where(['user_id' => $this->request->user()->id, 'exam_id' => (int)$id])-latest()-get();
$attempts = count($userExams);

if($attempts === 0)
{
    $userExam = UserExam::create(
        ['user_id' => $this->request->user()->id,
        'exam_id' => (int) $id,
        'attempt' => 1,
        'status' => 0
    ]);}
else
{
    if($userExams[0]->status === 0)
    {
        $userExam = $userExams[0];
        $userExamAnswers = UserExamAnswer::where('user_exam_id', $userExam->id)-get();
        $totalAnswered = count($userExamAnswers);
$$\text{temp} = \text{array}();$$
$$\text{foreach}($$\text{questions} \text{ as } \text{question})$$
$$\{$$
 $$\text{foreach}($$\text{userExamAnswers} \text{ as } \text{userExamAnswer})$$
 $$\{$$
 $$\text{if}($$\text{question}['id'] \text{ === }$$\text{userExamAnswer}-'$$\text{question_id})$$
 $$\{$$
 $$\text{question}['\text{selected_answer}'] =$$\text{userExamAnswer}-'$$\text{question_answer_id};$$
 $$\text{question}['\text{is_correct}'] =$$\text{userExamAnswer}-'$$\text{is_correct};$$
 $$\text{question}['\text{is_answered}'] =$$\text{userExamAnswer}-'$$\text{is_correct};$$
 $$\}$$
 $$\}$$
 $$\text{temp} = \text{arrayprepend}($$\text{temp}, $$\text{question});$$
 $$\}$$
 $$\text{questions} = \text{temp};$$
 $$\}$$
 $$\text{else}$$
 $$\{$$
 $$\text{userExam} = \text{UserExam::create([}$$
 $$'\text{user_id}' => \text{this}-'$$\text{request}->user()-'$$\text{id},$$
 $$'\text{exam_id}' => \text{(int)} \text{id},$$
 $$'\text{attempt}' => \text{attempts} + 1,$$
 $$'\text{status}' => 0$$
 $$]);$$
 $$\text{totalAnswered} = 0;$$
 $$\}$$
 $$\}$$
 $$\text{return view('app.pages.take-exam', ['exam' => $$\text{exam},$$
 $$'questions' => $$\text{questions}, 'total' => $$\text{total}, 'userExam' =>$$
 $$\text{userExam}, 'totalAnswered' => $$\text{totalAnswered]}));$$
 $$\}$$
public function home()
{
    $flashcards = Flashcard::where('user_id', Auth::id())->get();
    $exams = UserExam::where(['user_id' => Auth::id(), 'status' => 1])->latest()->get();
    $tempExams = array();
    $tempScores = array();
    $totalScores = 0;
    $totalExams = 0;
    $latestScore = 0;
    $latestExam = null;

    $scoresArray = array();
    $mean = 0;
    $median = 0;
    $sd = 0;

    foreach($exams as $key => $exam)
    {
        if($key == 0) {
            $latestScore = ($exam->score / count($exam->examQuestions)) * 100;
            $latestExam = $exam->exam->name . " - " . $exam->attempt;
        }

        if($key < 5) {
            array_push($tempExams, str_limit($exam->exam->name, 5) . " - " . $exam->attempt);
            array_push($tempScores, ($exam->score / count($exam->examQuestions)) * 100);
        }
        $totalScores += (($exam->score / count($exam->examQuestions)) * 100);
        array_push($scoresArray, ($exam->score / count($exam->examQuestions)) * 100);
    }
$totalExams++;  
}

if(count($exams) != 0)
    $mean = $totalScores / count($exams);

$sortedScores = $scoresArray;
sort($sortedScores);

$start = 0;
$end = 0;
$total = count($sortedScores);
if($total > 0)
{
    if(($total % 2) === 0)
    {
        $first = ($total/2) - 1;
        $end = ($total/2);
        $median = ($sortedScores[$first] + $sortedScores[$end])/2;
    }
    else
    {
        $median = $sortedScores[(floor($total/2)) - 1];
    }
}

$tempSD = 0;
foreach($scoresArray as $score)
{
    $tempSD += (($mean - $score) * ($mean - $score));
}

if($total > 0)
{
    $sd = $tempSD / $total;
}
return view(‘app.pages.home’, ['sd' => sqrt($sd), 'median' => $median, 'mean' => $mean, 'latestScore' => $latestScore, 'latestExam' => $latestExam, 'totalExams' => $totalExams, 'tempExams' => $tempExams, 'tempScores' => $tempScores, 'totalScores' => $totalScores, 'flashcards' => $flashcards]);
}

public function exams()
{
    $allowUserExams = AllowUserExam::where('user_id', $this->request->user()->id)->latest()->get();

    $oneAttemptExams = array();
    $practiseExams = array();
    $timedExams = array();
    foreach($allowUserExams as $allowUserExam)
    {
        if(empty($allowUserExam->exam->allow_retake) || $allowUserExam->exam->allow_retake === 0)
            $oneAttemptExams = array_prepend($oneAttemptExams, $allowUserExam->exam);
        else if(!empty($allowUserExam->exam->start_datetime))
            $timedExams = array_prepend($timedExams, $allowUserExam->exam);
        else
            $practiseExams = array_prepend($practiseExams, $allowUserExam->exam);
    }
    return view(‘app.pages.exams’, ['oneAttemptExams' => $oneAttemptExams, 'practiseExams' => $practiseExams, 'timedExams' => $timedExams]);
}
<?php
Route::group(['middleware' => ['checkrole']], function () {
    Route::prefix('admin')->group(function () {
        Route::get('/', 'AdminController@home')->name('admin.home');
        Route::get('/subjects', 'AdminController@subjects')->name('admin.subjects');
        Route::get('/professors', 'AdminController@professors')->name('admin.professors');
        Route::get('/professor/create', 'AdminController@newProfessor')->name('admin.new-professor');
        Route::get('/subject/create', 'AdminController@newSubject')->name('admin.new-subject');
        Route::post('/professor/create', 'AdminController@createProfessor')->name('admin.create-professor');
        Route::post('/subject/create', 'AdminController@createSubject')->name('admin.create-subject');
        Route::get('/subject/{id}/edit', 'AdminController@editSubject')->name('admin.edit-subject');
        Route::get('/professor/{id}/edit', 'AdminController@editProfessor')->name('admin.edit-professor');
        Route::delete('/professor/delete', 'AdminController@deleteProfessor')->name('admin.delete-professor');
        Route::delete('/subject/delete', 'AdminController@deleteSubject')->name('admin.delete-subject');
        Route::put('/professor/update', 'AdminController@updateProfessor')->name('admin.update-professor');
        Route::put('/subject/update', 'AdminController@updateSubject')->name('admin.update-subject');
        Route::put('/professor/update/password', 'AdminController@updateProfessorPassword')->name('admin.update-professor-password');
        Route::get('/subject/{id}/chapters', 'AdminController@chapters')->name('admin.chapters');
        Route::post('/subject/add-chapters', 'AdminController@addChapter')->name('admin.add-chapter');
    })->name('admin');
});
Route::delete('/chapter/delete', 'AdminController@deleteChapter')->name('admin.delete-chapter');
Route::get('/subject/chapter/{id}/edit', 'AdminController@editChapter')
->name('admin.edit-chapter');
Route::put('/subject/chapter/update', 'AdminController@updateChapter')
->name('admin.update-chapter');
Route::get('/dummy-data', 'FakerController@dummyData')
->name('adminfaker');

Route::prefix('professor')
->group(function () {
    Route::get('/', 'ProfessorController@home')
->name('professor.home');
    Route::get('/exams', 'ProfessorController@exams')
->name('professor.exams');
    Route::get('/questions', 'ProfessorController@questions')
->name('professor.questions');
    Route::get('/new-question', 'ProfessorController@newQuestion')
->name('professor.new-question');
    Route::post('/create-question', 'ProfessorController@createQuestion')
->name('professor.create-question');
    Route::get('/question/{id}', 'ProfessorController@question')
->name('professor.question');
    Route::get('/question/{id}/edit', 'ProfessorController@editQuestion')
->name('professor.edit-question');
    Route::put('/question/update', 'ProfessorController@updateQuestion')
->name('professor.update-question');
    Route::delete('/question/delete', 'ProfessorController@deleteQuestion')
->name('professor.delete-question');
    Route::get('/exams', 'ProfessorController@exams')
->name('professor.exams');
    Route::get('/new-exam', 'ProfessorController@newExam')
->name('professor.new-exam');
});
Route::post('/create-exam', 'ProfessorController@createExam')->name('professor.create-exam');
Route::get('/exam/{id}', 'ProfessorController@exam')->name('professor.exam');
Route::get('/exam/{id}/edit', 'ProfessorController@editExam')->name('professor.edit-exam');
Route::delete('/exam/delete', 'ProfessorController@deleteExam')->name('professor.delete-exam');
Route::get('/exam/{id}/add-questions', 'ProfessorController@addQuestions')->name('professor.add-questions');
Route::get('/exam/{id}/add-students', 'ProfessorController@addStudents')->name('professor.add-students');
Route::put('/exam/update', 'ProfessorController@updateExam')->name('professor.update-exam');
Route::get('/total-questions/{id}', 'ProfessorController@totalQuestionsJSON')->name('professor.total-questions');
Route::get('/exam-questions/{id}', 'ProfessorController@examQuestionsJSON')->name('professor.exam-questions');
Route::post('/add-question-exam', 'ProfessorController@addQuestionExam')->name('professor.add-question-exam');
Route::post('/remove-question-exam', 'ProfessorController@removeQuestionExam')->name('professor.remove-question-exam');
Route::get('/exam/students', 'ProfessorController@examUsers')->name('professor.exam-users');
Route::get('/total-students/{id}', 'ProfessorController@totalStudents')->name('professor.total-students');
Route::get('/exam-students/{id}', 'ProfessorController@examStudents')->name('professor.exam-students');
Route::post('/add-student-exam', 'ProfessorController@addStudentExam')->name('professor.add-student-exam');
Route::post('/remove-student-exam', 'ProfessorController@removeStudentExam')->name('professor.remove-student-exam');
Route::get('/statistics/{id}', 'ProfessorController@statistics')->name('professor.statistics');
Route::get('/summary/{user_exam_id}', 'ProfessorController@summary')->name('professor.summary');

Route::prefix('app')->group(function () {
    Route::get('/', 'AppController@home')->name('app.home');
    Route::get('/exams', 'AppController@exams')->name('app.exams');
    Route::get('/take-exam/{id}', 'AppController@takeExam')->name('app.take-exam');
    Route::post('/save-user-exam-answer', 'AppController@saveUserExamAnswer')->name('app.save-user-exam-answer');
    Route::put('/end-user-exam', 'AppController@endUserExam')->name('app.end-user-exam');
    Route::get('/summary/{user_exam_id}', 'AppController@summary')->name('app.summary');
    Route::get('/attempts/{exam_id}/{user_id}', 'AppController@attempts')->name('app.attempts');
    Route::post('/add-flashcard', 'AppController@addFlashcard')->name('app.add-flashcard');
    Route::get('/flashcards', 'AppController@flashcards')->name('app.flashcards');
});

Route::get('/', 'PublicController@landing')->name('landing');
Route::get('/login', 'PublicController@getLogin')->name('get-login');
Route::post('/do-login', 'PublicController@doLogin')->name('do-login');
Route::get('/logout', 'PublicController@logout')->name('logout');
Figure 14. Web.php (Routes)
APPENDIX B

PROJECT SCREENSHOTS
A web-based software which allows professors to give and students to take exams/assignments. This project focuses on making an effortless process for professors to make an exam, to grade exams for all students, and to create class statistic analysis reports about all exam-taking students. At the end of the exam, students can see their report with analytics based on topics in the exam. This web-based software will be built with Laravel PHP Framework for a backend, Bootstrap CSS Framework for a frontend, and MySQL for a database.

<table>
<thead>
<tr>
<th>Admin</th>
<th>Professor</th>
<th>Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manage professors and students</td>
<td>Manage questions</td>
<td>Take exam</td>
</tr>
<tr>
<td>Manage subjects and chapters</td>
<td>Manage exams</td>
<td>Exam summary</td>
</tr>
<tr>
<td>Creating a dummy data</td>
<td>Create exam from question pool</td>
<td>Flash cards</td>
</tr>
<tr>
<td></td>
<td>Practice exam</td>
<td>Exam Analysis</td>
</tr>
<tr>
<td></td>
<td>One-time exam</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Exam analyser</td>
<td></td>
</tr>
</tbody>
</table>

© Easy Exam - 2019

Figure 15. Home Screen

Figure 16. Login Screen
### Figure 17. Admin Dashboard

<table>
<thead>
<tr>
<th>Admin</th>
<th>Dashboard</th>
<th>Students</th>
<th>Professors</th>
<th>Subjects</th>
<th>Exam</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>50</td>
<td>3</td>
<td>5</td>
<td>15</td>
</tr>
</tbody>
</table>

### Figure 18. List of Subjects

- *quae quaeet*
  - Chapters: 451, 457
- *dissunt saepe*
  - Chapters: 451, 457
- *omnia melilia*
  - Chapters: 451, 457
- *ipsa labor*
  - Chapters: 451, 457
- *vemam auxuamnus*
  - Chapters: 451, 457
Figure 19. List of Chapters of Subject

Figure 20. List of Professors
Figure 21. Create New Professors

Figure 22. Create New Subject
Figure 23. Professor Dashboard

Figure 24. List of Questions
Figure 25. Create New Question

Figure 26. List of Exam
Figure 27. Create New Exam

Figure 28. Assign Questions to Exam
Figure 29. Assign Students to Exam

Figure 30. Exam Statistics – 1
Figure 31. Exam Statistics – 2

Figure 32. Student Dashboard
Figure 33. Exam List of Students

Figure 34. Attempt Report
### Chapter Breakdown

<table>
<thead>
<tr>
<th>#</th>
<th>Chapter</th>
<th>Total questions</th>
<th>% Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Prosectum Qui</td>
<td>9</td>
<td>20.00</td>
</tr>
<tr>
<td>2</td>
<td>Delorem Qui</td>
<td>10</td>
<td>16.00</td>
</tr>
<tr>
<td>3</td>
<td>Eb Modi</td>
<td>5</td>
<td>40.00</td>
</tr>
<tr>
<td>4</td>
<td>Dolor Dolorem</td>
<td>3</td>
<td>0.00</td>
</tr>
<tr>
<td>5</td>
<td>Et NIH</td>
<td>8</td>
<td>62.50</td>
</tr>
<tr>
<td>6</td>
<td>Quasi Officiis</td>
<td>6</td>
<td>16.67</td>
</tr>
<tr>
<td>7</td>
<td>Sepiunte Totam</td>
<td>4</td>
<td>30.00</td>
</tr>
<tr>
<td>8</td>
<td>Sit Aut</td>
<td>4</td>
<td>25.00</td>
</tr>
<tr>
<td>9</td>
<td>Voluptatem Ventris</td>
<td>1</td>
<td>0.00</td>
</tr>
</tbody>
</table>

### Question Breakdown

<table>
<thead>
<tr>
<th>#</th>
<th>Question excerpt</th>
<th>Chapter</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Et totum aut et voluptatem eum. Quia ut qua assemenda qui. Quia voluptatem in omnis laque non voluptatis aud. Et...</td>
<td>Voluptatem Ventris</td>
<td>✗</td>
</tr>
<tr>
<td></td>
<td>Placeat distinctio eum qui. Seppe raprodicande et officis. Eas officis vitae voluptatem odio. NIH dionemique non aspernatar quo. Totam voluptatem neciendae omnis eligendi velit quam misketar exercitationem.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Aut furo necessitatibus aut expedita. Quam reiciendae quo adipisci vitae temporeibus. Et autem est et nihil. Renum cense...</td>
<td>Sit Aut</td>
<td>✗</td>
</tr>
<tr>
<td></td>
<td>Show explanation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Voluptas maxime quem expedita omnis autem voluptate. Iusto soluta non animi id et in id. Igegnam non accusantium similique...</td>
<td>Sepiunte Totam</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Show explanation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Mob il saepe dolores vel nulla et sapiente. Nisi est dolores ut. Soluta quasi officia voluptatem in harum quia tempora...</td>
<td>Quasi Officiis</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Show explanation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Ocurrus asperiores architecto illo eum. Eas et et nihil unde dolore. Praesentium excepturi corporis voluptas reprehending...</td>
<td>Quasi Officiis</td>
<td>✗</td>
</tr>
<tr>
<td></td>
<td>Show explanation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Est quidem ipsum optio aspernatur expedita voluptas molestiae. Maxime eacuta rest qui. Voluptate et cupiditate animi mod...</td>
<td>Et NIH</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Show explanation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Quo ratione enim aut eveniet eum praesentium. Dicta quis dolores vitae omnis ex error. Temporibus facili ipsum volupt...</td>
<td>Dolor Delorem</td>
<td>✗</td>
</tr>
<tr>
<td></td>
<td>Show explanation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>fugia a et odio placeat et. Cumque ipsum eligendi et est ratione. Ieciosanding qui libera teneror fugiat. Non sit aut at...</td>
<td>Eb Modi</td>
<td>✗</td>
</tr>
<tr>
<td></td>
<td>Show explanation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 37. Flashcards

Figure 38. Taking Exam Screen
REFERENCES


