

بسم الله الرحمن الرحيم



Developing a Cloud-Based Application

BY

Aseel A. Al-saqqa 220190751

Basant E. Ayesh 220191028

Tasneem N. Abu Safia 220190492

Software Development Department

Faculty of Information Technology

Islamic University of Gaza

A Requirement for the Course Advanced Software Engineering (SDEV 4304)

Instructor: Dr. Rebhi S. Baraka

2022-2023

The Application URL:

[Train Management App](#)

The Github URL:

[Code in Github](#)

The Application data to login:

- **As Manager Account:**

admin@gmail.com
123456

- **As Trainee Account:**

Email: tasneemabusafyah@gmail.com
OR
Unique ID: 0356660609
123456789

- **As Advisor Account:**

Email: tsafia@students.iugaza.edu.ps
OR
Unique ID: 7859654697
123456789

Content table

Figures table 3

Abstract	4
1. Introduction	5
2. Cloud Software Application/Service Requirements	6
3. Software Architecture and Design.....	14
4. Used Cloud Services and its Interfaces	15
5. Implementation	16
6. Data.....	26
7. Architecture of the Used Cloud Platform.....	27
8. Deployment on the Platform.....	27
9. User Support	28
10. Conclusion.....	32
References	34

Figures table

Figure 1 use case system	8
Figure 2 use case Trainee	8
Figure 3 use case Manager	10
Figure 4 use case Advisor	12
Figure 5 NoSQL (firebase).....	26
Figure 6 Modern web app on Heroku	27
Figure 7 page of login to app	29
Figure 8 page of register trainee	29
Figure 9 notifications page after login -manager-.....	30
Figure 10 notification in manager page	30
Figure 11 registered account details	30
Figure 12 email details	30
Figure 13 page after login -Advisor-	31
Figure 14 page after login -trainee-.....	31
Figure 15 add task by user -advisor-.....	31
Figure 16 add meeting by user -trainee-.....	32
Figure 17 after logout return in login page	32

Abstract

The cloud-based IT training management application we developed aims to simplify the training process and facilitate effective communication among with four main components: Trainee, Manager, System, and Advisor. The application enables trainees to register, upload materials, apply for programs, and communicate with advisors. The Manager handles registration, authentication, data management, and billing, while the System provides essential functionalities like appointment management, notifications, and advertising. The application empowers advisors to effectively manage their assigned trainees, schedule meetings, and send notifications. It ensures seamless communication and coordination between trainees, managers, and advisors.

The application was developed using the Laravel framework, which provides a robust and scalable foundation for building web applications. Agile methodologies were adopted throughout the development process, enabling iterative and flexible development to meet evolving requirements. The application leverages both NoSQL and SQL databases for efficient and reliable data storage and retrieval. To enhance performance, scalability, and security, cloud-based enhancements were implemented. These enhancements take advantage of the cloud platform's infrastructure and services to optimize application performance, ensure reliable scalability, and enforce stringent security measures. The user interface of the application is designed to be user-friendly and intuitive, providing an easy-to-navigate experience for all stakeholders. The detailed documentation accompanying the application covers various aspects, including specifications, design decisions, implementation details, deployment instructions, and user guidelines. This documentation ensures comprehensive support and guidance for users, facilitating seamless adoption and utilization of the application.

1. Introduction

The purpose of this project was to develop a cloud-based IT training management application that facilitates the efficient management of trainees, advisors, and training programs in the field of information technology. The application provides a comprehensive solution for trainee registration, document uploading, training program selection, attendance tracking, meeting scheduling, and communication between trainees and advisors.

The project's primary objective was to create a user-friendly and efficient cloud application that streamlines the training management process. By leveraging cloud technology, the application offers scalability, accessibility, and data security to all users involved in the training program.

The Trainee component allows trainees to register, enter necessary information, and upload documents for registration. Trainees can upload personal files and training materials, apply for training programs, fill in training attendance forms, and request meetings with their advisors. The Manager component reviews training requests, generates unique trainee IDs, manages trainee accounts, handles billing issues, and has the authority to update data in the cloud. The System component serves as the backbone of the application, storing trainees' records, performing computations, and providing functionalities such as appointment management, notifications, emailing, searching, and advertising. The Advisor component allows advisors to register, manage their trainees, schedule meetings, send notifications or emails, and access relevant information.

Built on the Laravel framework, our application benefits from its robust and scalable foundation, allowing for the development of a high-quality web application. The adoption of Agile methodologies ensured an iterative and flexible approach, enabling us to adapt to evolving requirements and deliver an application that aligns with the needs of our stakeholders. To meet the project requirements, the application utilizes both SQL and NoSQL databases for data storage. It incorporates features such as meetings management, notifications, emailing, searching, and advertising. User activities are tracked and recorded, and analytics tools are implemented to monitor system activities and generate insights. The application boasts a user-friendly GUI tailored to the needs of managers, advisors, and trainees.

Overall, our cloud-based IT training management application offers a powerful solution for managing training programs, promoting effective communication, and ensuring a streamlined training experience for trainees, managers, system administrators, and advisors.

2. Cloud Software Application/Service Requirements

The cloud-based IT training management application we have developed is a comprehensive solution aimed at simplifying the training process and facilitating effective communication among trainees, managers, system administrators, and advisors.

User Stories:

1. For Trainee Registration:

As a trainee, I want to be able to easily register for IT training programs online.

Acceptance Criteria:

The registration process should be user-friendly and intuitive.

Trainees should be able to provide necessary information, such as their personal details and preferred training programs.

Upon accept registration, trainees should receive a confirmation notification.

2. For Advisor Registration:

As an advisor, I want to register and provide the necessary information for access to the application.

Acceptance Criteria:

User-friendly advisor registration process

Ability to enter personal information during registration

Validation checks for accurate and complete information

Confirmation email with login credentials upon successful registration.

3. For Trainee Management:

As an advisor, I want to manage my assigned trainees.

Acceptance Criteria:

Access to a dashboard or interface for trainee management

Ability to view trainee profiles and track their progress

Update trainee information and provide feedback

Efficient search and filtering capabilities for finding specific trainees

4. For Meeting Request:

As a trainee, I want to request meetings with advisors to discuss my training progress and goals.

Acceptance Criteria:

Trainees should be able to submit meeting requests through the application.

Trainees should be able to provide details about the purpose of the meeting and preferred date and time.

Advisors should receive the meeting requests and be able to accept, reschedule, or decline them.

Trainees should receive notifications about the status of their meeting requests.

5. For Manager Approval:

As a manager, I want to review and approve trainee registrations for IT training programs.

Acceptance Criteria:

Managers should have access to a dashboard or interface where they can review trainee registration requests.

Managers should be able to approve or reject registrations based on eligibility criteria and program availability.

Trainees should receive notifications about the status of their registration requests.

6. For Generate Unique Trainee IDs:

As a manager, I want to generate unique trainee IDs for authorized user registration.

Acceptance Criteria:

When a trainee successfully registers for an IT training program, a unique trainee ID should be generated and assigned to their account.

The trainee ID should be used as a unique identifier for the trainee throughout the system.

The trainee ID should be associated with the trainee's registration and used for tracking their progress and interactions within the application.

7. For Manage Accounts:

As a manager, I want to update data in the cloud, manage trainee accounts, and handle billing.

Acceptance Criteria:

Managers should have the ability to update and manage trainee data stored in the cloud-based database.

Managers should be able to modify trainee information, such as personal details, training program selection, and progress.

Managers should have access to tools and functionalities to handle billing processes, such as generating invoices, tracking payments, and managing financial transactions related to trainee accounts.

8. For Track and Record User Activities:

As a manager, I want to track and record user activities.

Acceptance Criteria:

The system should have the capability to track and record user activities within the application.

User activities, such as trainee registrations, program selections, meeting requests, and progress updates, should be logged and stored in a secure and auditable manner.

The recorded user activities should be available for analysis and reporting purposes, enabling managers to gain insights into trainee behavior, program popularity, and overall system usage.

Use Cases:

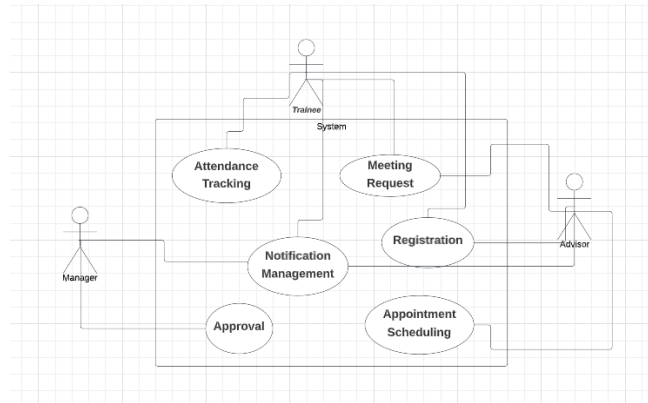


Figure 1 use case system

❖ For the Trainee component:

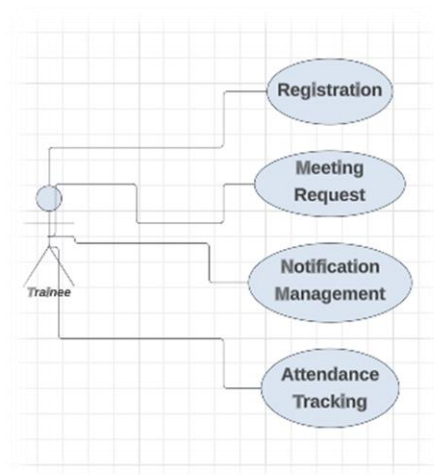


Figure 2 use case Trainee

1. Trainee Registration

Identifier	TR 01
Actors	Trainee, System
Goal	Trainee registers for an IT training program
Precondition	None
Postcondition	Trainee is successfully registered for the selected training program
Main Scenario	1. Trainee accesses the registration page.
	2. Trainee enters their personal information (name, email, contact details, etc.).
	3. Trainee submits the registration form.
	4. Manager validates the information provided.
	5. Manager accepts the trainee's registration request.
	6. System sends a confirmation email to the trainee with their ID and password.
	7. Trainee receives the confirmation email.
	8. Trainee logs in to their account using the provided user ID and password.

2. Meeting Request

Identifier	MR 01
Actors	Trainee, Advisor
Goal	Trainee requests a meeting with the assigned advisor
Precondition	Trainee is registered and logged into the system
Postcondition	Meeting request is successfully submitted to the advisor
Main Scenario	1. Trainee navigates to the meeting request section
	2. Trainee selects the assigned advisor
	3. Trainee specifies the purpose, date, and time for the meeting
	4. Trainee submits the meeting request
	5. Advisor receives the meeting request notification
	6. Advisor reviews the request and accepts or suggests an alternative time
	7. Trainee receives the response from the advisor

3. Attendance Tracking

Identifier	AT 01
Actors	Trainee, System
Goal	Trainee records their attendance for training sessions
Precondition	Trainee is registered for a training program
Postcondition	Attendance is successfully recorded for the trainee
Main Scenario	1. Trainee accesses the attendance tracking section
	2. Trainee views the list of upcoming training sessions
	3. Trainee selects the appropriate training session
	4. Trainee confirms their attendance for the session
	5. The system updates the attendance record for the trainee

4. Notification Management

Identifier	NM 01
Actors	Trainee, Advisor, Manager
Goal	Users manage their notification preferences
Precondition	The user is registered and logged into the system
Postcondition	Notification preferences are successfully updated
Main Scenario	1. User accesses the notification management settings
	2. User selects the desired notification preferences (e.g., email, in-app notifications)
	3. User saves the updated notification settings
	4. Trainee confirms their attendance for the session
	5. System updates the user's notification preferences

5. Trainee Document Upload

Identifier	TR 02
Actors	Trainee, System
Goal	Trainee uploads training-related documents
Precondition	Trainee is registered for a training program
Postcondition	Documents are successfully uploaded and associated with the trainee's profile

Main Scenario	1. Trainee accesses the document upload feature.
	2. Trainee selects the documents to upload (e.g., certificates, resumes, application forms).
	3. Trainee uploads the selected documents.
	4. System validates the document formats and sizes.
	5. System associates the uploaded documents with the trainee's profile.
	6. Trainee receives a confirmation message of successful document upload.

❖ *For Manager component:*

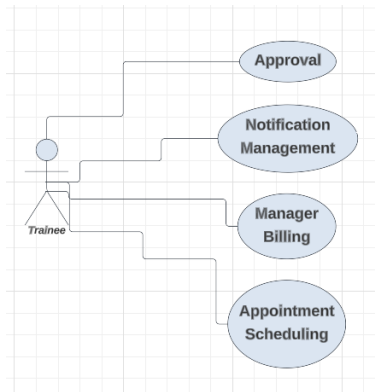


Figure 3 use case Manager

6. Manager Approval

Identifier	MA 01
Actors	Manager, System
Goal	Manager approves or rejects a trainee's training request
Precondition	Trainee has submitted a training request
Postcondition	Trainee's training request is approved or rejected
Main Scenario	1. Manager receives a notification about a pending training request
	2. Manager accesses the request details
	3. Manager reviews the trainee's information and training program
	4. Manager approves or rejects the training request
	5. System updates the request status accordingly
	6. Trainee receives a notification about the approval or rejection decision

7. Appointment Scheduling

Identifier	AS 01
Actors	Advisor, System
Goal	Advisor schedules a meeting with a trainee
Precondition	Advisor and trainee have an active relationship
Postcondition	Meeting is scheduled between the advisor and trainee
Main Scenario	1. Advisor accesses the appointment scheduling feature
	2. Advisor selects the trainee with whom they want to schedule a meeting
	3. Advisor chooses a suitable date and time for the meeting
	4. Advisor confirms the appointment details
	5. System schedules the meeting and sends a notification to the trainee
	6. Trainee receives the notification and acknowledges the meeting

8. Manager Billing

Identifier	MA 02
Actors	Manager, System
Goal	Manager generates and manages billing for trainees
Precondition	Trainees have completed training programs or incurred fees
Postcondition	Trainees' billing information is accurately generated and managed
Main Scenario	1. Manager accesses the billing management feature.
	2. Manager selects the trainees for whom billing needs to be generated.
	3. System calculates the fees or charges based on completed training programs or other factors.
	4. Manager reviews the generated billing information for accuracy.
	5. Manager sends the billing details to trainees via email or notifications.
	6. Trainee receives a confirmation message of successful document upload.

❖ *For System component:*

9. Communication via Notifications

Identifier	CN 01
Actors	Trainee, Advisor, System
Goal	Trainee and advisor communicate through notifications
Precondition	Trainee and advisor have an active relationship
Postcondition	Trainee and advisor exchange messages through notifications
Main Scenario	1. Trainee or advisor composes a message within the application
	2. System sends a notification to the intended recipient
	3. Recipient receives the notification and accesses the message
	4. Recipient composes a reply and sends it through the application
	5. System delivers the reply as a notification to the original sender
	6. Sender receives the notification and reads the reply

10. System Analytics

Identifier	SY 01
Actors	System
Goal	System generates analytics and reports on training program performance
Precondition	Training programs have been conducted and completed
Postcondition	Analytics and reports provide insights into the performance and effectiveness of training programs
Main Scenario	1. System collects and aggregates relevant data from completed training programs.
	2. System analyzes the data to generate key performance indicators (KPIs) and metrics.
	3. System generates reports and visualizations based on the analyzed data.
	4. System presents the analytics and reports to managers.
	5. Managers or administrators can access and review the analytics and reports.
	6. System updates the analytics and reports periodically to reflect the latest data.

❖ *For Advisor component:*

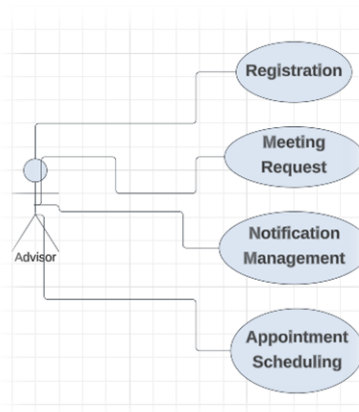


Figure 4 use case Advisor

11. Advisor Registration

Identifier	TR 01
Actors	Trainee, System
Goal	Trainee registers for an IT training program
Precondition	None
Postcondition	Trainee is successfully registered for the selected training program
Main Scenario	1. Advisor accesses the registration page.
	2. Advisor enters the required personal information (e.g., name, email, password).
	3. Advisor submits the registration form.
	4. Manager validates the information provided.
	5. Manager approves the advisor's registration request.
	6. System sends a confirmation email to the advisor with their ID and password.
	7. Advisor receives the confirmation email.
	8. Advisor logs in to their account using the provided user ID and password.

12. Advisor Trainee Management

Identifier	AD 01
Actors	Advisor, System
Goal	Advisor manages the trainees assigned to them
Precondition	Advisor has trainees assigned to their profile
Postcondition	Advisor effectively manages and supports the assigned trainees
Main Scenario	1. Advisor accesses the trainee management feature.
	2. Advisor views the list of trainees assigned to their profile.
	3. Advisor can filter or search for specific trainees based on criteria.
	4. Advisor can view trainee details, progress, and performance.
	5. Advisor can provide feedback and guidance to trainees through the system.
	6. Advisor can schedule meetings or send notifications to trainees when necessary.

Requirements specifications:

❖ Trainee Functionality:

- Trainees should be able to view available IT training programs.
- Trainees should be able to register for training programs of their choice.
- Trainees should be able to upload required documents and materials.
- Trainees should be able to request meetings with advisors.
- Trainees should receive notifications regarding their registration status, meeting requests, and program updates.

❖ Manager Functionality:

- Managers should have access to a dashboard for reviewing trainee registration requests.
- Managers should be able to approve or reject trainee registrations based on eligibility criteria and program availability.
- Managers should be able to generate unique trainee IDs for authorized user registration.
- Managers should be able to update trainee data and manage trainee accounts.
- Managers should be able to handle billing processes.

❖ Advisor Functionality:

- Advisors should be able to register and provide the necessary information for access to the application.
- Advisors should be able to manage their assigned trainees and track their progress.
- Advisors should be able to accept, reschedule, or decline meeting requests from trainees.
- Advisors should be able to communicate with trainees through notifications or emails.
- Advisors should be able to send notifications and emails to trainees as needed.

❖ System Functionality:

- The system should store and retrieve user data securely.
- The system should provide appointment management features for trainees and advisors.
- The system should send notifications and emails to users based on specific events and actions.
- The system should have advertising functionality to promote training programs.
- The system should record user activities for analytics and auditing purposes.

❖ Performance and Security:

- The application should be responsive and provide a seamless user experience.
- The application should be scalable to handle a large number of users and data.
- The application should implement appropriate security measures to protect user data and ensure data privacy.
- Regular backups of data should be performed to prevent data loss.

3. Software Architecture and Design

application architecture

The architecture of the cloud-based IT training management application is designed to simplify the training process and facilitate effective communication among stakeholders. The application consists of four main components: Trainee, Manager, System, and Advisor. Each component serves specific roles and functionalities to ensure seamless coordination and interaction.

❖ **Front-end/UI**

The front-end of the application encompasses the user interface (UI) components visible to users. It is designed to be user-friendly and intuitive, providing an easy-to-navigate experience. The UI includes screens, forms, and interactive elements that allow users to register, upload materials, apply for programs, and communicate with advisors.

❖ **Back-end**

The back-end of the application is built using the Laravel framework, which provides a robust and scalable foundation for web application development. The back-end follows the Model-View-Controller (MVC) architectural pattern, separating the application logic into three components

❖ **Databases**

The application leverages both NoSQL and SQL databases for efficient and reliable data storage and retrieval. NoSQL databases(firebase), are used for CV file . SQL databases(MySQL), are used for structured all information. The choice of database depends on the specific requirements of each component of the application.

❖ **Cloud-based Enhancements**

To enhance performance, scalability, and security, the application incorporates cloud-based enhancements. These enhancements take advantage of the cloud platform's infrastructure and services to optimize application performance, ensure reliable scalability, and enforce stringent security measures. Examples of cloud-based enhancements include cloud hosting services, load balancing, auto-scaling, content delivery networks (CDNs), and security services like firewalls, encryption, and access control.

❖ **Agile Development Methodology**

The development process of the application follows agile methodologies. Agile promotes collaborative teamwork, continuous feedback, and frequent iterations, allowing for faster development and adaptation to changing needs. The use of agile methodologies enables the development team to deliver a high-quality application that meets evolving requirements.

Functional components

Each component of the application can be implemented as separate microservices, allowing for scalability, modularity, and ease of maintenance. the main functional components of the application:

- **Trainee Component:**
Handles trainee registration, document uploads, training program selection, meeting requests, and attendance forms.
- **Manager Component:**
Manages training requests, trainee accounts, billing, and updates data in the cloud. Handles authentication and authorization for trainee login.
- **Advisor Component:**
Handles advisor registration, trainee management, meeting scheduling, and communication with trainees.
- **System Component:**
Stores trainees' records, performs necessary computations, and provides functionalities such as notifications, emailing, searching, and advertising. Utilizes SQL and NoSQL databases for data storage.

4. Used Cloud Services and its Interfaces

This cloud-based IT training management app uses Heroku Cloud Platform as a Service (PaaS) to host and deploy the app, the service is paid (not free). Heroku has been a leader in the cloud computing industry since its development in June 2007, initially supporting the Ruby programming language and later expanding to include Java, Node.js, Scala, Clojure, Python, PHP, and Go. This makes Heroku a multilingual platform, offering features and support for building, running, and scaling applications across multiple programming languages.

Interfaces provided by Heroku:

Application Hosting: Heroku serves as the hosting environment for the application. It provisions the necessary resources, including computing power, storage, and networking capabilities, to ensure the availability and performance of the application. The deployed application is accessible to users over the internet.

Database Integration: Heroku offers managed database services, and in this case, Heroku Postgres is used as the underlying SQL database for the application. The application interfaces with the Heroku database service to store and retrieve data securely. Heroku Postgres provides reliability, scalability, and efficient data management, contributing to the overall functionality of the application.

By leveraging the Heroku cloud platform, the application benefits from its robust infrastructure and the ability to build, run, and scale applications seamlessly across various programming languages. The interfaces with Heroku's application hosting and managed database services ensure the smooth deployment and reliable data storage and retrieval for the IT training management application.

5. Implementation

Implementation details need to be explained, Codes, libs, etc.

Stage1: Code Development

[Code in Github](#)

```
public function sendEmail(Request $request)
{
    $meetingId = $request->meeting_id;
    $emailContent = $request->email_content;
    $meeting = Meeting::findOrFail($meetingId);
    $email = $meeting->trainee->user->email;
    $data = [
        'emailContent' => $emailContent,
        'meetingDate' => $meeting->datetime,
        'advisor' => Auth::user()->name,
    ];
    Mail::to($email)->send(new MeetingEmail($data));
    return response()->json(['msg' => 'Done']);
}

public function getNotifications()
{
    $notifications = Notification::latest()->limit(10)->orderBy('created_at', 'desc')->get();

    return response()->json([
        'notifications' => $notifications,
    ]);
}

public function markAsRead(Request $request, $id)
{
    $notification = Notification::findOrFail($id);
    $notification->read_at = Carbon::now();
    $notification->save();
    if (!$request->ajax()) {
        if (strcmp($notification->type, string2: "register_Advisor") == 0) {
            return redirect()->route( route: 'advisors.show', json_decode($notification->data, associative: true));
        } else if (strcmp($notification->type, string2: "register_Trainee") == 0) {
            return redirect()->route( route: 'trainees.show', json_decode($notification->data, associative: true));
        } else if (strcmp($notification->type, string2: "assignCourse") == 0) {
            return redirect()->route( route: 'courses.show', json_decode($notification->data, associative: true));
        }
    }
}

public function pushNotification($advisor_id, $course_id)
{
    $notification = Notification::create([
        'type' => 'assignCourse',
        'notifiable_type' => 'App\User',
        'notifiable_id' => $advisor_id,
        'data' => json_encode([
            'course_id' => $course_id,
            'title' => 'New Message',
            'body' => 'New Course #' . $course_id . ' Added to You',
        ]),
    ]);

    $pusher = new Pusher([
        'auth_key' => '1e58abe6fe45f3bd2e73',
        'secret' => 'c4f5a1132840e7111aba',
        'app_id' => '16075',
        'cluster' => 'ap3'
    ]);

    $pusher->trigger( channels: 'advisor', event: 'notify-advisor', [
        'title' => 'New Message',
        'body' => 'New Course #' . $course_id . ' Added to You',
        'notification_id' => $notification->id,
        'notifiable_id' => $advisor_id,
        'course_id' => $course_id,
    ]);
}
```



```

    }

    public function getAllTask(Request $request)
    {
        $courseId = $request->courseId;
        $course = Course::findOrFail($courseId);
        $courseName = $course->name;
        if ($request->ajax()) {
            $data = Task::with(['course', 'advisor'])->where('course_id', $courseId)->get();
            return DataTables::of($data)
                ->addIndexColumn()
                ->addColumn('action', function ($task) {
                    $buttons = '<div class="btn-group" role="group">';
                    if (Auth::user()->guard == 'advisor') {
                        $buttons .= '<a href="' . route('tasks.edit', $task) . '" class="btn btn-1';
                        $buttons .= '<a class="mainDelete btn btn-light-danger" data-id="' . $task->id . '"><i class:
                    }
                    $buttons .= '<a href="' . route('tasks.submissions', $task) . '" class="btn bt
                    $buttons .= '</div>';
                    return $buttons;
                })
                ->addColumn('file', function ($task) {
                    if ($task->file) {
                        return '<div class="file"><img alt="Task file thumbnail" data-id="' . $task->id . '"></div>';
                    }
                });
        }
    }

    public function getAllTrainee(Request $request)
    {
        $courseId = $request->courseId;
        $course = Course::findOrFail($courseId);
        $courseName = $course->name;
        if ($request->ajax()) {
            $trainees = $course->trainees()
                ->wherePivot('status', 'active')
                ->with('user')
                ->get();
            return DataTables::of($trainees)
                ->addIndexColumn()
                ->addColumn('action', function ($trainee) {
                    $buttons = '
                    <div class="btn-group" role="group">
                        <a href="' . route('trainees.show', $trainee) . '" class="btn btn-light-primary"><i c
                    if (Auth::user()->guard == 'manager') {
                        if ($trainee->status == 'inactive') {
                            $buttons .= '<a class="traineeActive btn btn-light-success" data-id="' . $t
                        } else {
                            $buttons .= '<a class="traineeDeActive btn btn-light-danger" data-id="' . $
                        }
                    }
                    $buttons .= '<a class="mainDelete btn btn-light-danger" data-id="' . $trainee->
                });
        }
    }

    //accept Trainee in course
    public function active($course_id, $trainee_id)
    {
        $course = Course::findOrFail($course_id);

        if ($course->num_trainee >= $course->capacity) {
            // toastr()->error('The course has reached its maximum capacity.');
```

```

    }

    public function store(Request $request)
    {
        $validatedData = Validator::make($request->all(), [
            'name' => 'required|string|max:255|unique:fields',
        ]);
        if ($validatedData->fails()) {
            return response()->json(['error' => $validatedData->errors()->first()]);
        }

        $field = new Field;
        $field->name = $request->name;
        $field->save();

        // Return a response or redirect as needed
        return response()->json(['msg' => 'Field created successfully']);
    }

    /**
     * Display the specified resource.
     */
    public function show(string $id)
    {
        //
    }
}

public function traineeRequests(Request $request)
{
    if ($request->ajax()) {
        $data = Trainee::where(['status' => 'inactive'])->with(['user' => function ($q) {
            $q->where(['status' => 'inactive']);
        }])->get();
        return DataTables::of($data)
            ->addIndexColumn()
            ->addColumn( name: 'action', function ($trainee) {
                $buttons = '
                <div class="btn-group" role="group">
                <a href="' . route( name: 'trainees.show', $trainee) . '" class="btn btn-light-primary"><i c

                if ($trainee->status == 'inactive') {
                    $buttons .= ' <a class="traineeActive btn btn-light-success" data-id="' . $trainee->id . '">Active
                } else {
                    $buttons .= ' <a class="traineeDeActive btn btn-light-danger" data-id="' . $trainee->id . '">DeActive
                }

                $buttons .= ' <a class="mainDelete btn btn-light-danger" data-id="' . $trainee->id . '">Delete
            </div>';
            return $buttons;
        })
    }
}

public function join(Request $request){
    $course = Course::findOrFail($request->course_id); $trainee = Auth::user()->trainee;
    // Check if trainee already joined the course
    if ($trainee->courses->contains($course)) {
        return back()->with('error', 'You have already joined this course.');
```

```

<script>

  Pusher.logToConsole = true;
  var pusher = new Pusher('1e58abe6fe45f3bd2e73', {
    cluster: 'ap3',
    forceTLS: true
  });
  @if(auth()->user()->guard === 'manager')
  var channel1 = pusher.subscribe('newRegister');
  channel1.bind('new-register', function (data) {
    console.log(data);
    $('#messagePusher .modal-body').html(data.body);
    $('#messagePusher .modal-title').html(data.title);
    $('#messagePusher').modal('show');
    updateUnreadCount();
    var audio = document.getElementById('notification-sound');
    audio.play();

    $('#messagePusher #messagePusherForm').attr('action', function () {
      var actionUrl = '{{ route("notificationsRead", ":notificationId") }}';
      actionUrl = actionUrl.replace(':notificationId', data.Notification_id);
      return actionUrl;
    });

    $('#showRegisterPusher').on('click', function (e) {
      e.preventDefault(); // Prevent the default form submission
    });
  });

  script -> callback for channel1.bind()

@elseif
  var authID = {{ Illuminate\Support\Facades\Auth::id() }};
  @if(auth()->user()->guard === 'advisor')
  var channel2 = pusher.subscribe('advisor');
  channel2.bind('notify-advisor', function (data) {
    if (authID === data.notifiable_id) {
      console.log(data);
      $('#messagePusher .modal-body').html(data.body);
      $('#messagePusher .modal-title').html(data.title);
      $('#messagePusher').modal('show');
      updateUnreadCount();
      var audio = document.getElementById('notification-sound');
      audio.play();

      $('#messagePusher #messagePusherForm').attr('action', function () {
        var actionUrl = '{{ route("notificationsRead", ":notificationId") }}';
        actionUrl = actionUrl.replace(':notificationId', data.Notification_id);
        return actionUrl;
      });

      $('#showRegisterPusher').on('click', function (e) {
        e.preventDefault(); // Prevent the default form submission
        $('#messagePusherForm').submit();
      });

      setTimeout(function () {
        }, 20000);
    }
  });
  @endif
$(function () {
  $('#messagePusher #closePusher').on('click', function (e) {
    $('#messagePusher').modal('hide');
  });
})

function updateUnreadCount() {
  jQuery.ajax({
    url: "{{ route('notifications.count') }}",
    method: "GET",
    success: function (data) {
      $('#unreadCount').text(data);
    },
    error: function (xhr, status, error) {
      console.error(error);
    }
  });
}

</script>
script -> callback for channel2.bind()

```

```

54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
table.on('click', 'advisorDeActive', function (e) {
    e.preventDefault();
    var id = $(this).data('id');
    var URL = "{{ route('advisorDeActive', 'x') }}";
    var new_url = URL.replace('x', id);
    $.ajax({
        url: new_url,
        dataType: "json",
        type: 'GET',
        success: function (html) {
            console.log(html);
            table.DataTable().ajax.reload();
        }
    });
});
table.on('click', 'advisorActive', function (e) {
    e.preventDefault();
    var id = $(this).data('id');
    var URL = "{{ route('advisorActive', 'x') }}";
    var new_url = URL.replace('x', id);
    $.ajax({
        url: new_url,
        dataType: "json",
        type: 'GET',
    });
});

```

```

public function uploadFilesFireBase($request)
{
    $firebaseCredentialsPath = storage_path(env( key: 'FIREBASE_CREDENTIALS_PATH'));
    $storage = new StorageClient([
        'projectId' => 'trainmanagement-b559e',
        'keyFilePath' => $firebaseCredentialsPath,
    ]);

    $bucket = $storage->bucket( name: 'trainmanagement-b559e.appspot.com');

    // Store CV Files
    if ($request->hasFile('file')) {
        $cvFile = $request->file('file');
        $cvPath = 'Files/' . ($request->name ?? $request->title) . '.' . time() + rand(1, 10000000)
            | '.' . $cvFile->getClientOriginalName();
        $bucket->upload(
            file_get_contents($cvFile),
            [
                'name' => $cvPath,
            ]
        );
        return $cvPath;
    }

    return '';
}

```

\App\Http\Traits > FileUploadTrait > uploadFilesFireBase()

```

}

function getUploadedFireBase($filePath)
{
    if (!empty($filePath)) {
        $firebaseCredentialsPath = storage_path(env( key: 'FIREBASE_CREDENTIALS_PATH'));
        $storage = new StorageClient([
            'projectId' => 'trainmanagement-b559e',
            'keyFilePath' => $firebaseCredentialsPath,
        ]);

        $bucket = $storage->bucket( name: 'trainmanagement-b559e.appspot.com');

        // Generate the signed URL for the file
        $object = $bucket->object($filePath);
        $url = $object->signedUrl(now()->addHour());

        return $url;
    }

    return null;
}

```

\App\Http\Traits > FileUploadTrait > uploadFilesFireBase()

```

namespace App\Http\Middleware;

use ...

class ManagerMiddleware
{
    public function handle(Request $request, Closure $next)
    {
        if (Auth::check() && Auth::user()->guard === 'manager') {
            return $next($request);
        }

        return redirect()->route( route: 'login')->with('error', 'Try Again!');
    }
}

```

\App\Http\Middleware > ManagerMiddleware

```

        $taskSubmission->delete();
        return redirect()->back()->with('msg', 'Task submission deleted successfully');
    }

    public function updateMark(Request $request, TaskSubmission $submission)
    {
        $request->validate([
            'mark' => 'required|integer',
        ]);

        $submission->mark = $request->input( key: 'mark');
        $submission->status = 'completed';
        $submission->save();

        return response()->json(['message' => 'Submission mark updated successfully']);
    }
}

```

\App\Http\Controllers\AdvisorControllers > TaskSubmissionController

```

    public function create()
    {
        $courses = Course::where([
            'advisor_id' => Auth::user()->advisor->id,
        ]->where('end_date', '>=', date( format: 'Y-m-d'))->get();
        return view( view: 'layouts.tasks.create', compact( var_name: 'courses'));
    }

    public function store(CreateTaskRequest $request)
    {
        DB::transaction(function () use ($request) {
            $path = $this->uploadFilesFireBase($request);
            $request['advisor_id'] = Auth::user()->advisor->id;
            $task = Task::create(array_merge($request->all(), [
                'file' => $path,
            ]));
        });
        return redirect()->route( route: 'tasks.index')->with('msg', 'Task created successfully');
    }

    public function show($id)
    {
        $task = Task::findOrFail($id);
    }
}

```

\App\Http\Controllers\AdvisorControllers > TaskController

```

public function postLogin(Request $request)
{
    $credentials = $request->validate([
        'userName' => 'required|string',
        'password' => 'required|string',
    ]);

    $field = filter_var($credentials['userName'], filter: FILTER_VALIDATE_EMAIL) ? 'email' : 'unique_';

    $loginData = [
        $field => $credentials[$field],
        'password' => $credentials['password'],
        'status' => 'active',
    ];

    if (Auth::attempt($loginData)) {
        $request->session()->regenerate();

        return redirect()->route('home');
    } else {
        return back()->with('error', 'These credentials do not match our records.');
```

```

public function postTrainee(TraineeRegistrationRequest $request)
{
    DB::beginTransaction();

    $user = User::create([
        'name' => $request->name,
        'email' => $request->email,
        'password' => Hash::make($request->password),
        'guard' => 'trainee',
        'status' => 'inactive',
    ]);

    // Upload files
    $uploadedFiles = $this->uploadFilesFireBase($request);

    $trainee = Trainee::create([...]);
    $fields = $request->input(key: 'fields');
    $trainee->fields()->attach($fields);
    $this->pushNotificationManager($trainee->id, type: 'Trainee');
    DB::commit();

    // Redirect to a success page or perform any additional actions
    return redirect()->route('login')->with('success', 'Trainee registered successfully. Please login.');
```

AuthController.php

```

public function changePass(Request $request)
{
    $user = Auth::user();
    $request->validate([
        'password' => 'required|min:8|confirmed',
    ]);

    $user->password = Hash::make($request->password);
    $user->save();

    return redirect()->back()->with('msg', 'Password changed successfully.');
```

```

public function forgotPassword(Request $request)
{
    $validatedData = Validator::make($request->all(), [
        'email' => 'required|email|exists:users',
    ]);
    if ($validatedData->fails()) {
        return back()->with('error', 'You should Register.');
```

AuthController.php

```

public function pushNotificationManager($register_id, $type)
{
    $manager = User::where(['guard' => 'manager'])->first();
    $notification = Notification::create([
        'type' => 'register_' . $type,
        'notifiable_type' => 'App\User',
        'notifiable_id' => $manager->id,
        'data' => json_encode([
            'register_id' => $register_id,
            'title' => 'New Message',
            'body' => 'New ' . $type . ' Register #' . $register_id,
        ]),
    ]);

    $pusher = new Pusher([
        'auth_key' => '1e58abe6fe45f3bd2e73',
        'secret' => 'c4f5a1132840e7111aba',
        'app_id' => '160',
        'cluster' => 'ap3'
    ]);
    $pusher->trigger([
        'channels' => 'newRegister',
        'event' => 'new-register',
        'title' => 'New Message',
        'body' => 'New ' . $type . ' Register #' . $register_id,
        'Notification_id' => $notification->id,
    ], [
    ]);
}

public function updateProfile(Request $request)

```

\App\Http\Controllers\Auth > AuthController > postTrainee()

BillingController.php

```

public function store(Request $request)
{
    $validator = Validator::make($request->all(), [
        'visa' => 'required|unique:billings,visa|regex:/^4[0-9]{3}\s?[0-9]{4}\s?[0-9]{4}\s?[0-9]{4}',
        'amount' => 'required|numeric',
        'payment_date' => 'required|date',
        'cvc' => 'required|digits:4'
    ]);

    if ($validator->fails()) {
        return back()->withErrors($validator)->withInput();
    }

    Billing::create([
        'trainee_id' => Auth::user()->trainee->id,
        'visa' => $request->visa,
        'amount_due' => $request->amount,
        'payment_date' => $request->payment_date,
        'cvc' => $request->cvc,
    ]);

    return redirect()->route('route: 'billings.create')->with('success', 'Billing created successful');
}

```

\App\Http\Controllers > BillingController > create()

```

}

public function courseAdvisors()
{
    $traineeId = auth()->user()->trainee->id;

    $advisors = CourseTrainee::where('trainee_id', $traineeId)
        ->join('advisors', 'course_trainee.advisor_id', '=', 'advisors.id')
        ->select('advisors.*')
        ->with('advisor.user')
        ->pluck('id');

    $advisors = Advisor::with('relations: 'user')->whereIn('column: 'id', $advisors)->get();
    return response()->json($advisors);
}

```

Stage2: Uploading to the Cloud & Database Integration

Manual deploy

Deploy the current state of a branch to this app.

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more](#)

Choose a branch to deploy

main

Deploy Branch

Receive code from GitHub

Build main

Release phase

Deploy to Heroku

Your app was successfully deployed.

View

heroku.com

Blogs

Careers

Documentation

Support

Terms of Service

Privacy

Cookies

© 2023 Salesforce.com

training-management-app

Profile

web: vendor/bin/heroku-php-apache2 public/

Commit Changes

Changes

Profile

Before Commit

After Commit

Commit Message

add Profile to deploy commit

Diff

Salesforce Platform

HEROKU

Jump to Favorites, Apps, Pipelines, Spaces...

GitHub Tasneem-Abu-Sufian/TrainCloudProject

Overview

Resources

Deploy

Metrics

Activity

Access

Settings

Dynos

This app has no process types yet

Add-ons

Quickly add add-ons from Elements

There are no add-ons for this app

Estimated Monthly Cost

\$0.00

Salesforce Developers

HEROKU

Products

Marketplace

Pricing

Documentation

Support

More

Search Elements

Log in

Sign up

Add-ons

Buttons

Buildpacks

Heroku Postgres

Reliable and powerful database as a service based on PostgreSQL. Starting at \$5/mo.

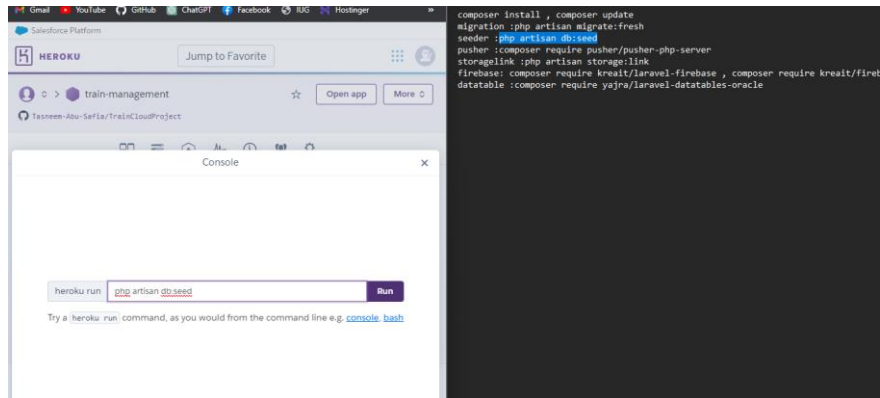
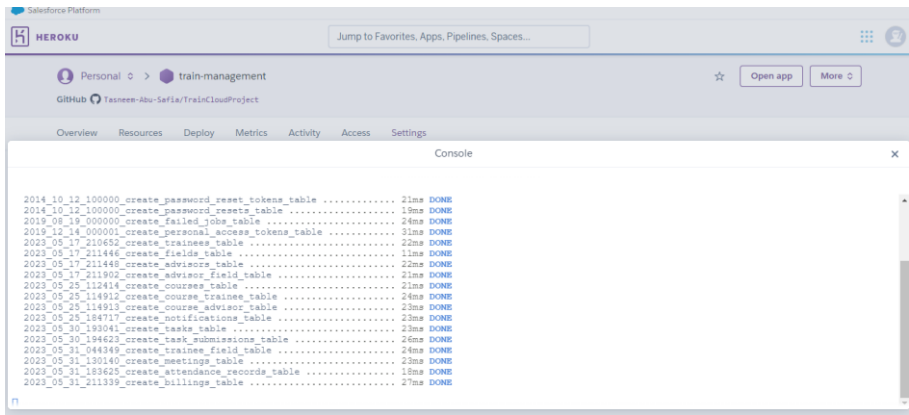
Install Heroku Postgres

Quick Links

Shareable Details
















Add-on Category

Data Stores



DATABASE_URL	postgres://xtylzybtkvydm:712697f9cf6c02a	✎ ✕
DB_CONNECTION	pgsql	✎ ✕
DB_DATABASE	dbb5pbupu8f514	✎ ✕
DB_HOST	ec2-107-21-67-46.compute-1.amazonaws.com	✎ ✕
DB_PASSWORD	712697f9cf6c02a8b8817cb4ff9ef3eef3e2494e3	✎ ✕
DB_PORT	5432	✎ ✕
DB_USERNAME	xtylzybtkvydm	✎ ✕
FIREBASE_CREDENTIALS_PATH	firebase/trainmanagement-firebase.json	✎ ✕
FIREBASE_PROJECT_ID	trainmanagement-b559e	✎ ✕
FIREBASE_STORAGE_BUCKET	trainmanagement-b559e.appspot.com	✎ ✕




MAIL_FROM_ADDRESS	"TrainSystem@gmail.com"	✎ ✕
MAIL_FROM_NAME	"\${APP_NAME}"	✎ ✕
MAIL_HOST	smtp.gmail.com	✎ ✕
MAIL_MAILER	smtp	✎ ✕
MAIL_PASSWORD	svcpcehlyqzpeph1	✎ ✕
MAIL_PORT	465	✎ ✕
MAIL_USERNAME	tasneemabusafyah@gmail.com	✎ ✕
PUSHER_APP_CLUSTER	ap3	✎ ✕
PUSHER_APP_ID	1607529	✎ ✕
PUSHER_APP_KEY	1e58abe6fe45f3bd2e73	✎ ✕

MAIL_USERNAME	tasneenabusafyah@gmail.com	 
PUSHER_APP_CLUSTER	ap3	 
PUSHER_APP_ID	1687529	 
PUSHER_APP_KEY	1e58abe6fe45f3bd2e73	 
PUSHER_APP_SECRET	c4f5a1132848e7111aba	 
PUSHER_PORT	443	 
PUSHER_SCHEME	https	 
KEY	VALUE	

Salesforce Platform

HEROKU Jump to Favorites, Apps, Pipelines, Spaces...

Deployment method

 Heroku Git Use Heroku CLI
 GitHub **Connected**
 Container Registry Use Heroku CLI

App connected to GitHub


Code diffs, manual and auto deploys are available for this app.

Connected to [Tasneem-Abu-Safa/TrainCloudProject](#) by [Tasneem-Abu-Safa](#) Disconnect...

Releases in the [activity feed](#) link to GitHub to view commit diffs

Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

 You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions [here](#).

Enable automatic deploys from GitHub


Every push to the branch you specify here will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch is always in a deployable state and any tests have passed before you push. [Learn more](#)

Choose a branch to deploy



6. Data

I used SQL (Heroku Postgres) for all the regular data of the system components and NoSQL (Google Firebase) for saving files uploaded by users.



Firebase

Project Overview 

Project shortcuts

-  **Storage**
-  Authentication

What's new

-  Extensions NEW
-  Functions NEW


Product categories


- Build
- Release and monitor
- Analytics

Spark
No cost \$3/month Upgrade

TrainManagement

Storage

Files Rules Usage  Extensions NEW

[gs://trainmanagement-b559e.appspot.com](#) > Files Upload file 





<input type="checkbox"/>	Name	Size	Type	Last modified
<input type="checkbox"/>	 Advisor.169288567.test.docx	13.32 KB	application/vnd.open-officedocument.word	1 Jun 2023
<input type="checkbox"/>	 Tasneem.1690517447.test.docx	13.32 KB	application/vnd.open-officedocument.word	1 Jun 2023
<input type="checkbox"/>	 aseel.1690200333.cloud app requirements.pdf	209.69 KB	application/pdf	1 Jun 2023
<input type="checkbox"/>	 aseel.1692498942.cloud app requirements.pdf	209.69 KB	application/pdf	1 Jun 2023

Figure 5 NoSQL (firebase)

7. Architecture of the Used Cloud Platform

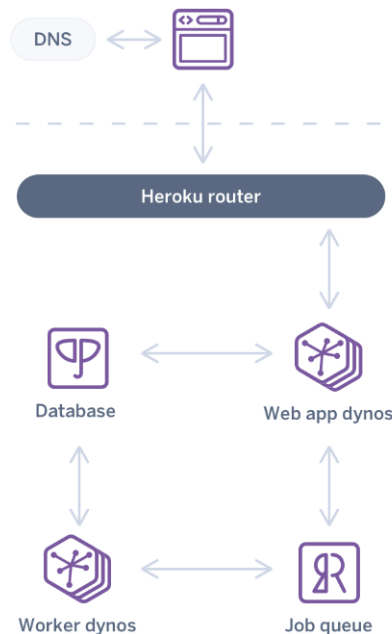


Figure 6 Modern web app on Heroku

The target platform deployment environment for the cloud-based IT training management application is Heroku. Heroku is a cloud platform as a service (PaaS) that provides a robust infrastructure for hosting and deploying web applications. It supports multiple programming languages, making it a versatile choice for developers.

In the Heroku deployment environment, the application architecture follows a typical PaaS model. The application code, along with its dependencies and configurations, is packaged into a deployable unit called a "slug." The slug is then deployed to the Heroku platform, where it is executed and managed by the Heroku runtime environment.

Heroku follows a distributed architecture that includes multiple layers and components to ensure efficient application execution and scalability. These layers include:

Dynos: Dynos are lightweight Linux containers that run the application code. Heroku dynamically scales the number of dynos based on the application's resource needs and incoming traffic.

Routing: Heroku's routing layer acts as a load balancer, distributing incoming requests across the available dynos. It ensures that requests are efficiently routed to the appropriate dyno handling the application.

Heroku Postgres: Heroku's managed relational database service, Heroku Postgres, is used for data storage in the application. It provides reliability, scalability, and data durability.

8. Deployment on the Platform

To deploy the IT training management application on the Heroku platform, the following steps were followed:

Application Configuration: The application was prepared for deployment by ensuring all necessary dependencies and configurations were in place. This includes specifying the required runtime environment, defining environment variables, and setting up database connections.

Repository Setup: The application code, along with any relevant files, was pushed to a Git repository. This repository served as the source of truth for the application's codebase.

Deployment Process: Using the Heroku Command Line Interface (CLI), the application was deployed to the Heroku platform. The deployment process involved creating a new Heroku app, linking it to the Git repository, and initiating the deployment command.

Build and Release: Heroku automatically builds the application using the specified build pack, which determines the necessary runtime and dependencies. The build process generates a deployable slug.

Application Execution: Heroku creates and manages dynos to run the application code. The routing layer distributes incoming requests to the appropriate dyno based on the specified routing rules.

After completing the execution steps in the uploading to the cloud phase, we wrote these commands:

composer install

migration :php artisan migrate:fresh

seeder :php artisan db:seed

pusher :composer require pusher/pusher-php-server

storagelink :php artisan storage:link

firebase: composer require kreative/laravel-firebase ,

composer require kreative/firebase-php

datatable :composer require yajra/laravel-datatables-oracle

9. User Support

User Roles and Permissions:

❖ **Managers:**

Managers have the highest level of permissions and access.

Functionalities:

Approve trainee and advisor registrations for IT training programs.

Authenticate and manage trainee and advisor accounts.

Update trainee and advisor data and profiles.

Manage billing and payment processes.

Generate reports and analytics.

Access administrative features and settings.

Communicate with trainees and advisors.

Assign advisors to trainees.

❖ **Trainees:**

Trainees have limited permissions within the application.

Functionalities:

Register for IT training programs.

Upload material.

Apply for specific training programs.

Request meetings with assigned advisors.

Receive notifications.

❖ **Advisors:**

Advisors have specific permissions related to managing trainees.

Functionalities:

Register as an advisor and provide necessary information.

Manage assigned trainees and their progress.

Schedule and conduct meetings with trainees.

Communicate with trainees through notifications or emails.

Provide guidance and support to trainees.

Access relevant trainee data and profiles.

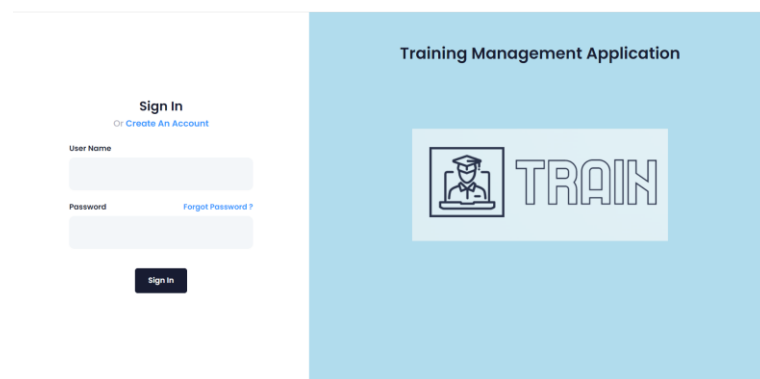


Figure 7 page of login to app

Figure8 page of register trainee

When a new account is created, the manager receives a notification to verify the validity of the user's data (figure 20), and then activates the account or not (figure 21).

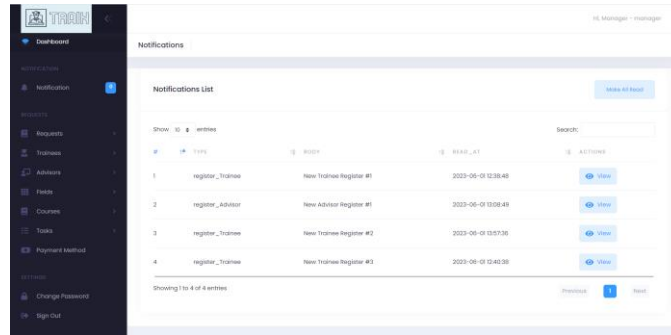


Figure 9 notifications page after login -manager-

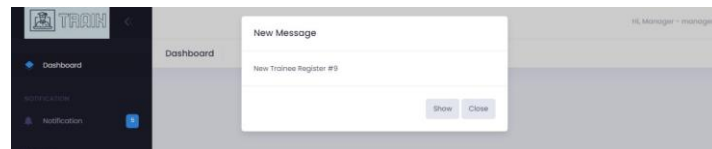


Figure 10 notification in manager page

Unique Id:

Phone:

Address:

Degree:

Fields:

- Web Development

Status:

Files: Open CV

Activate Back

Figure 11 registered account details

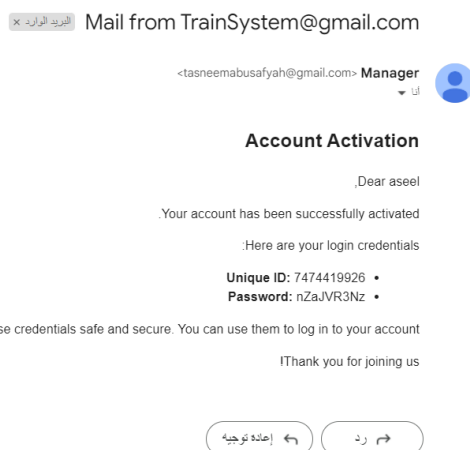


Figure 12 email details

The user receives an e-mail confirming the acceptance of the registration, which contains information to enter the application, his id and password

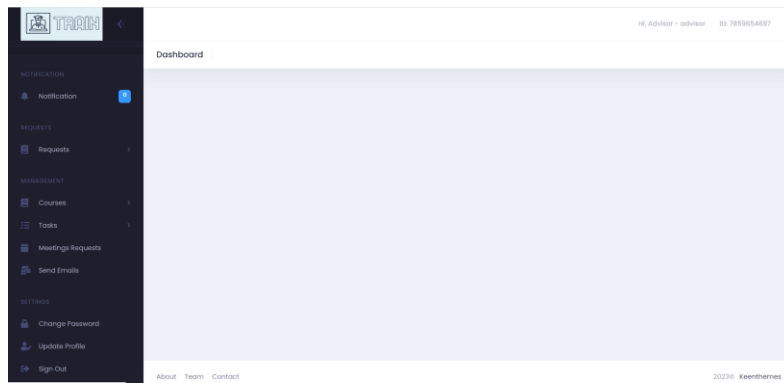


Figure 13 page after login -Advisor-

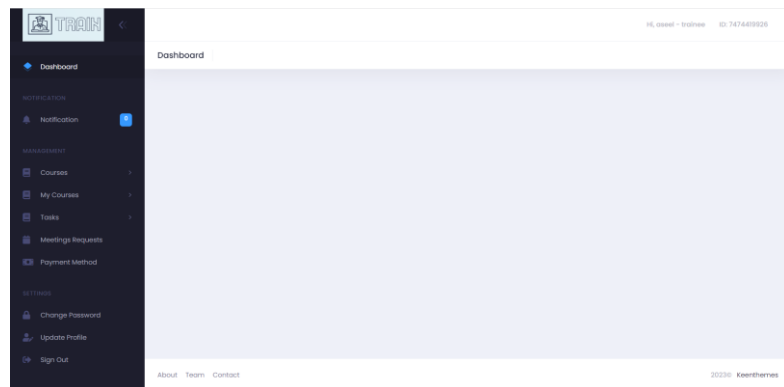


Figure 14 page after login -trainee-

The user logs in with the information that he received and uses the application with ease.

Task

Course*	Title*
<input type="text" value="Select Course"/>	<input type="text" value="Enter Title"/>
Start Date*	End Date*
<input type="text" value="mm/dd/yyyy"/>	<input type="text" value="mm/dd/yyyy"/>
Description*	Mark
<input type="text" value="Type a Description"/>	<input type="text" value="Enter Mark"/>
File	
<input type="button" value="Choose File"/> No file chosen	
<input type="button" value="Save"/> <input type="button" value="Reset"/> <input type="button" value="Back"/>	

Figure 15 add task by user -advisor-

Figure 16 add meeting by user -trainee-



Figure 17 after logout return in login page

10.Conclusion

In conclusion, the cloud-based IT training management application is designed to simplify the training process and facilitate effective communication among trainees, managers, and advisors. The application's architecture consists of four main components: Trainee, Manager, System, and Advisor. Each component plays a crucial role in ensuring seamless coordination and efficient management of training programs.

The Trainee component enables trainees to register, upload training materials, apply for programs, and communicate with advisors. The Manager component handles registration, authentication, data management, and billing processes. The System component provides essential functionalities such as appointment management, notifications, and advertising. The Advisor component empowers advisors to effectively manage their assigned trainees, schedule meetings, and send notifications.

The application's architecture is built on the Laravel framework, following the Model-View-Controller (MVC) pattern, which separates the application logic into models, views, and controllers. It leverages both NoSQL and SQL databases for efficient and reliable data storage and retrieval. Cloud-based enhancements are implemented to enhance performance, scalability, and security, taking advantage of the cloud platform's infrastructure and services.

The user interface of the application is designed to be user-friendly and intuitive, providing an easy-to-navigate experience for all stakeholders. Agile methodologies were adopted during the development process, allowing for iterative and flexible development to meet evolving requirements. The comprehensive documentation accompanying the application provides support and guidance for users, facilitating seamless adoption and utilization of the application.

In summary, the cloud-based IT training management application provides a robust and scalable solution for managing training programs, streamlining communication, and facilitating effective collaboration among trainees, managers, and advisors. The application's architecture, user interface design, and cloud-based enhancements work together to deliver a seamless and efficient experience for all users involved in the training process.

References

<https://medium.com/@agavitalis/how-to-send-an-email-in-laravel-using-gmail-smtp-server-53d962f01a0c>

<https://pusher.com/docs/>

<https://laravel.com/docs/10.x>

<https://www.youtube.com/watch?v=DV4YICpZ5FE>

<https://www.youtube.com/watch?v=skGZ8laUQco>

[Beyond Web and Worker: Evolution of the Modern Web App on Heroku | Heroku](#)