

SULEYMAN DEMIREL UNIVERSITY
ENGINEERING FACULTY

CSS 348- Advanced Database Management Systems

Project report- “**Activ-Monitoring**”

Checked by: Yesdaulet Izenov

Prepared by: Aruzhan Serikbayeva

Aimzhan Sytdykova

Assylan Aitmambet

2020

I. Introduction

1.1 Problem definition

There are companies that develop software and, of course, in such enterprises, you always need control over projects for high-quality and productive work. And often the problem is to discover early on that the activity will take longer than expected. If you discover the problem late, the activity can go astray, be costly to fix, and soon interfere with many other activities.

1.2 Scope

In our documentation we have:

- Introduction to our project;
- Functional requirements;
- System and User Interfaces;
- System Design(drawn via Figma);
- Product Functions;
- Team;
- Use case Diagram;
- Development process;
- Problems with implementation;

1.3 Overview

To begin with, we didn't know about the Laravel framework and made each page separately in PHP. There were also difficulties when connecting MySQL to the laptop when connecting, errors with the test came out. In general, to make it easier to work, we used the following life hacks from the Visual Studio Editor, we needed applications such as: **draw.io integration**(to make it easier to work with tables), **MySQL** (to make it easier to connect to the database and manage them).

II. Overall description

2.1. Product Perspective

Our web application is designed to detect that a project may take longer than planned at an early stage, so we can help the developer who is currently carrying out this activity, for example, by allowing an experienced expert to evaluate the work and offer recommendations. In addition, project managers and department managers typically oversee several activities. And with the help of our web application, they can get a good overview of the projects that they are particularly closely following.

2.1.1 System interfaces - Web application

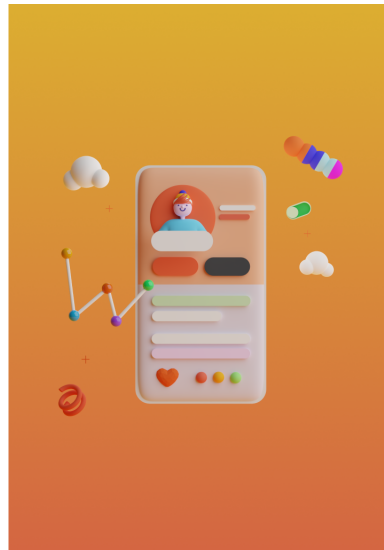
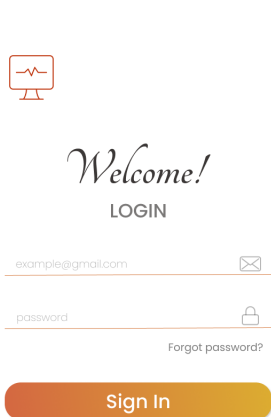
- Windows
- macOS
- Linux

2.1.2 User Interfaces

Web Application: Activ-Monitoring

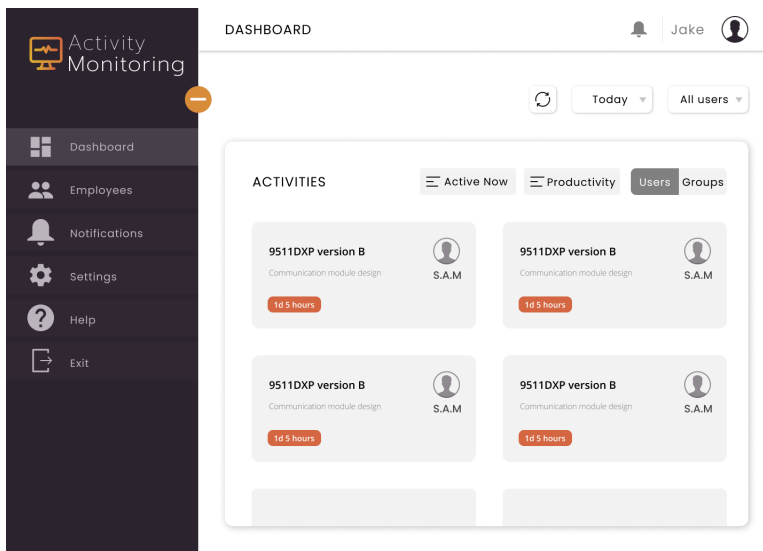
We used Figma for this part of the project:

<https://www.figma.com/file/yPvI8DwAi7EuTrqult5bbh/UI?node-id=0%3A1>



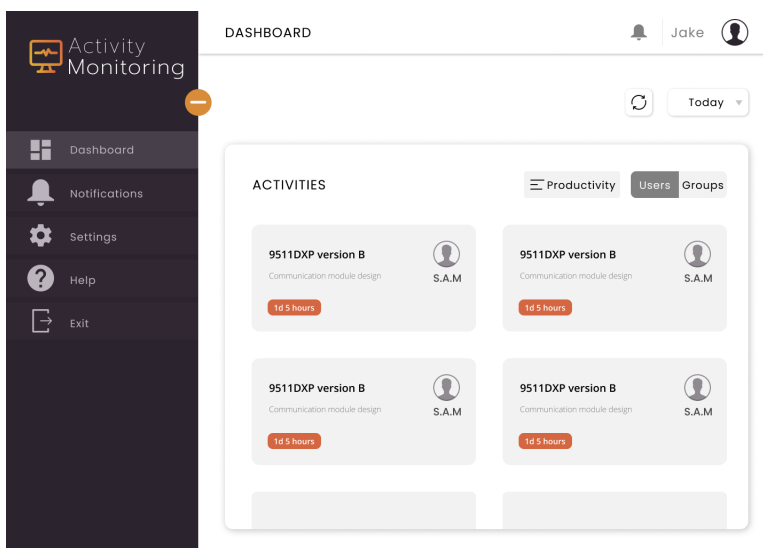
Login Page

This is the first page, the login page, that the user sees when opening the site. Here we have enabled the user to log in. He can write his username and password here, and we have also done the registration part for possible new users. But in order for the employee to log in, he will need to enter the correct username and password, otherwise notifications will pop up that have the wrong username or password. And we also introduced such a typical function as password masking. And only after that, you can already log in.



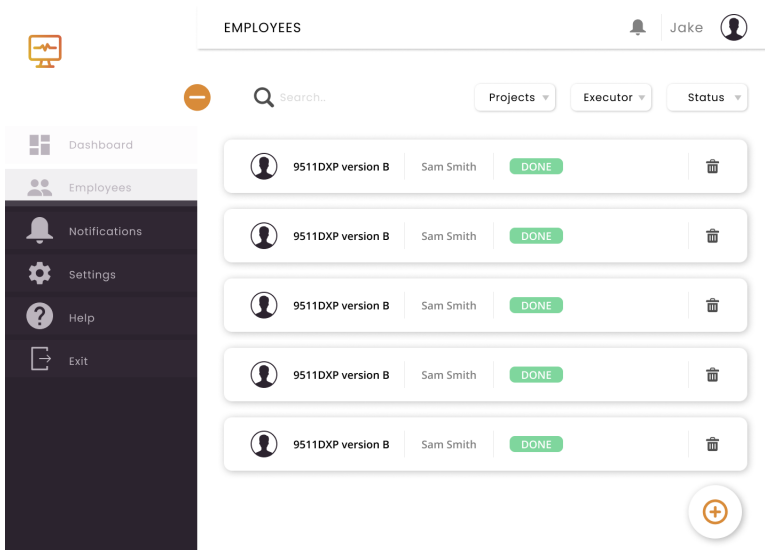
Dashboard (Manager)

After successful authorization, we made it so that our system redirects the user to the main page of the manager or employee, depending on which username the user enters. If the employee is a project manager, then the system accordingly translates to a page with an Activity Board. And the Project Manager will have access to functions such as viewing active developers, sorting by developer performance, also sorting by task date, updating content, and so on.



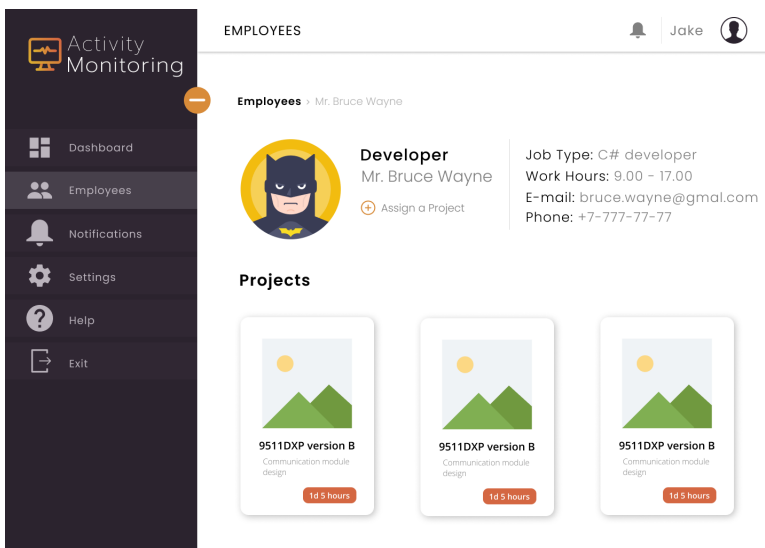
Dashboard (Developer)

After successful authorization, if the employee is a developer, then the next page is a page where functions are possible, such as sorting actions by their performance, by the date of completion of the task, on this page we added a button to update the content. We also made it so that the developer could see their tasks and the deadline for their completion on the activity board itself.



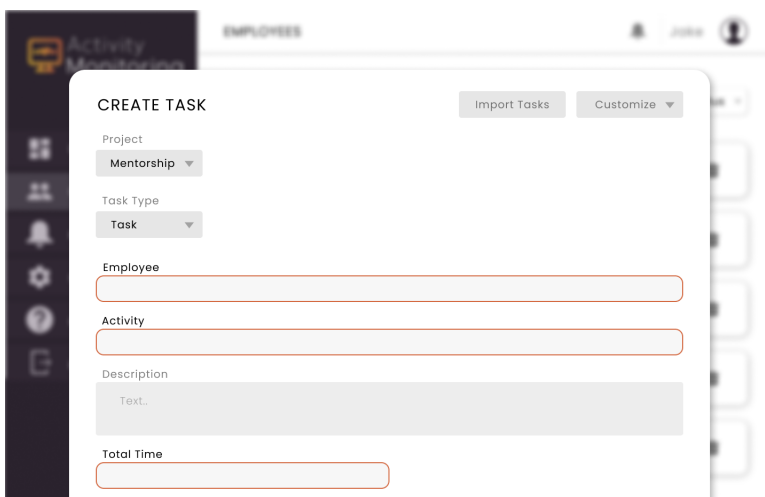
Employees (Manager)

In this page, we have provided a system for project managers that provides an "Employees" tab for better project management and employee monitoring. For example, in this page there is a search, with this function PM can search for all developers available in the company, we also added "Projects" and with this function PM can specify the search for projects, and with the help of "Performer" PM finds the right developers. And by "Status", the PM can see the status of the employee's task at the moment, and so on. We tried to add more of these features to make it easier to find activity.



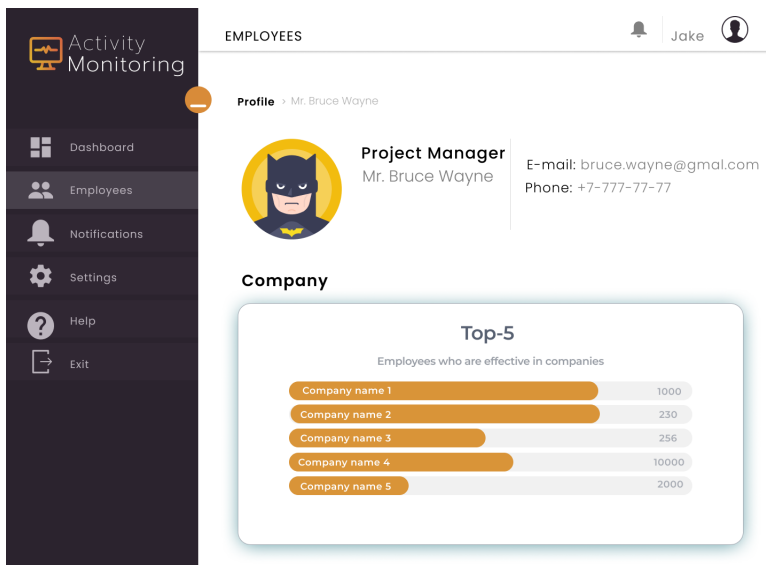
Employee (Manager)

On this page, we gave the Project Manager control over the Developers. The manager can view information such as the employee's role in the project, full name, working hours, mailing address, and phone number. About the types of activities that the employee performs. What else is this developer working on? Also, the project manager is given the opportunity to create projects and assign a responsible developer by type of activity using the "Assign Project" button.



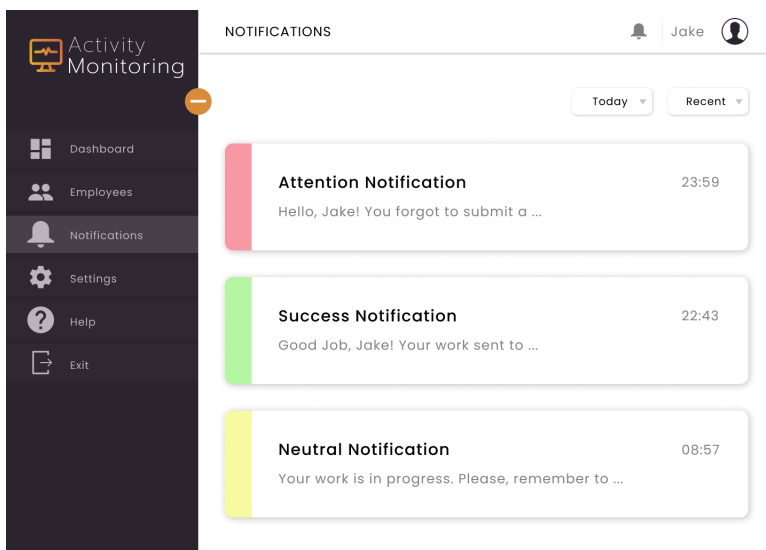
Create Task (Manager)

In the next page, we have developed a pop-up window where the manager can easily fill out a form in order to set an employee a task or create an event. There are buttons such as "Project", which allows you to select a specific project, the "Task" button-selects a specific type of task in the project, also in the employee selection panel you need to enter the initials of the employee, "Activity" as it is clear from the name of this type of activity, then "Description" of the project. The "Total time" string for the total time to complete a specific task, and so on.



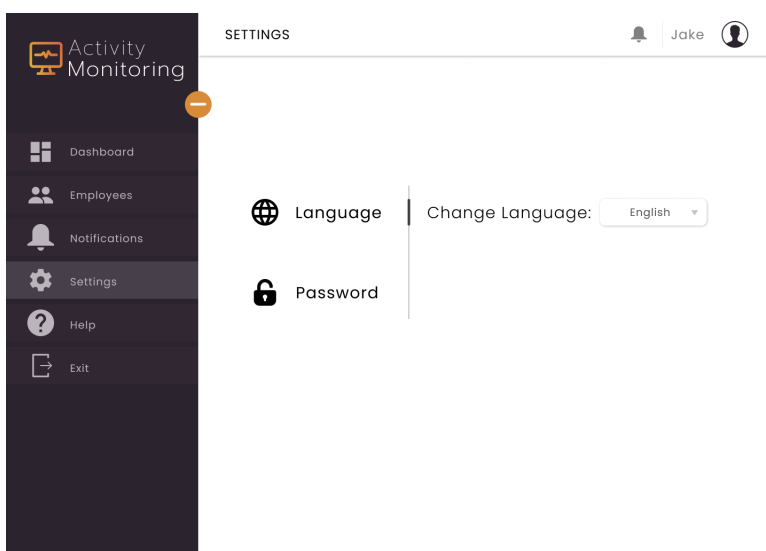
Manager Profile

Also, we created a profile page for project managers with their own project statistics and her/his own information like name, surname, email etc.



Notifications

We also created a notification page for developers and project managers. On this page, you can see the drop-down tab mentioned above, meaning you can use this button to sort information or notifications by date. The next thing the user can see is the same drop-down tab that you can use to sort recent or long-standing notifications. Next, the page displays the notifications themselves, that is, a list of them with different colors. Red color - the employee missed the deadline or did not do something important on the project. Green - successfully completed tasks. And the yellow ones are just reminders about the progress of the project, so that the user remembers about it.



Settings

And of course, as without the settings page. Here we have added features by changing the language on the page and change the password if necessary.

Activity Monitoring

HELP

Jake

Please send your inquiry, if you are faced with difficulties.

Your Name:

Your E-Mail:

Theme:

Question Type:

General

Technical

Message:

Send

Cancel

Help Page

And the last thing that we did is a page with the help of. Here the user can get help or answers to their questions.

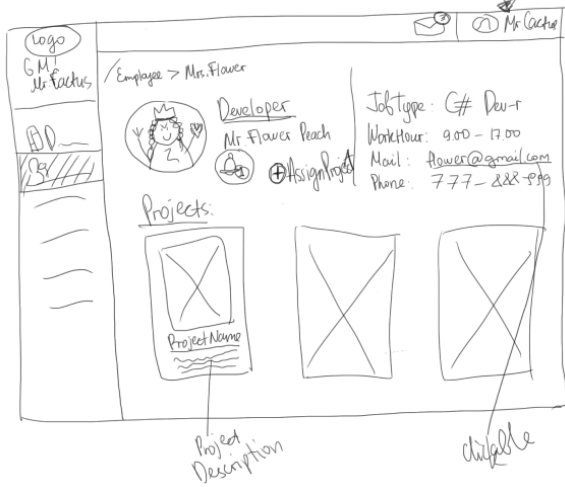
1. In the "Your Name" text box, he must enter his full name.
2. Then there is a field for assigning his mail.
3. Below is the field for the subject of the question.
4. In addition, he can choose the type of question by clicking on one of the two boxes.
5. Write in the form of messages about the problem itself.
6. Send by clicking on the "Send" button.
7. Or if he changes his mind, then click on the "Cancel" button, and he will return to the previous page.

2.2 System design

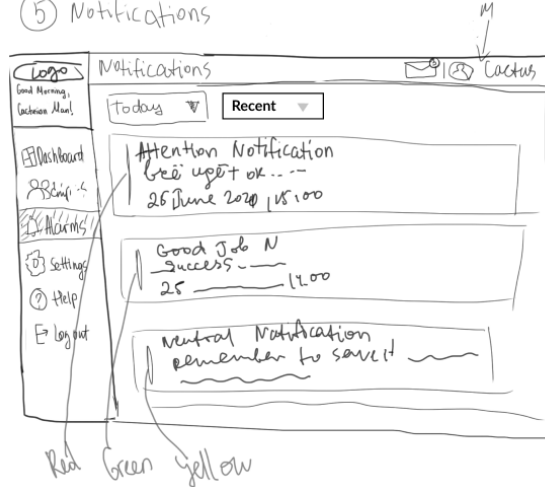
We also used Figma for this part. We will attach screenshots of some UX drafts below.



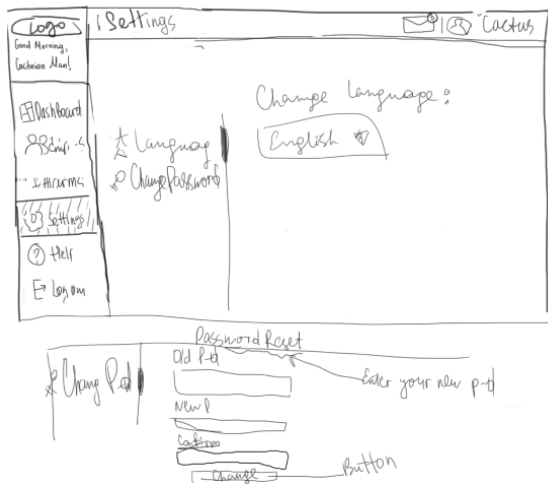
④ Emp Page (from Manager)



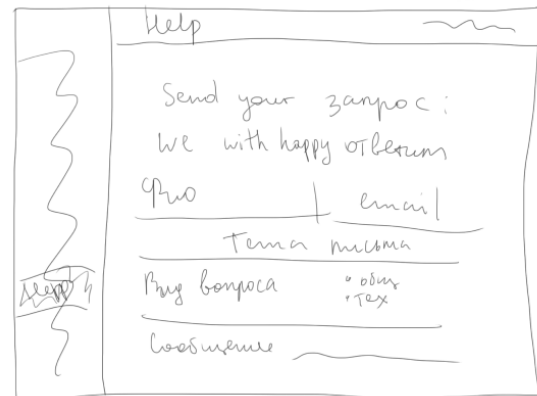
⑤ Notifications



⑥ Settings



⑦ Help



Further on, we have the experience of interaction (UX). In this part, we spent much more time than on the user interface, as we thought about how to make our interface as simple and convenient as possible. I think that this part can be called a problem, or rather here we are faced with some problems. Initially, we started doing everything from scratch and made a gross mistake, since the bike was happening. Then we decided to look for similar sites and templates. That's when it became easier for us to continue working, as we understood where to start and were able to visually imagine how the site should look and how convenient it should be. One such saito was ActiveTrack. We took it as an example, because it was most similar to our idea. But we did not take the entire ActiveTrack site, but tried to add something of our own, for example, notifications about the developer's report, and also tried to make the part where the manager needs to set the activity more convenient. Thus, we have a roughly user-friendly design.

2.3 Product functions

Our web-app functions:

1. Login to the system
2. Adaptation to mobile devices
3. Seven-digit numbers for each employee
4. Redirects the user to the main page when logging in to the account
5. For PM: redirects to the activity distribution page

6. For developers: redirect to the main page
7. The ability for the user to fill in and send the form with the time spent, as well as calculate the remaining time.
8. The system should allow the user to fill out and submit a form with the time spent, as well as calculate the remaining time.
9. Reminder to make a report - "the function of the rest"
10. When the remaining time allotted for work is 0, the "reminder function" should inform you about it
11. The project manager has the option to select a developer account and assign a project
12. The administrator should be able to see all employees and their working hours
13. The system language is English.
14. The system must reflect the new and changed product description within x minutes after the database is updated by the product owner.
15. The "I forgot my password " section, which sends a link to the verified email
16. If one developer helped another developer, or if more than one employee is involved in one activity, the "transfer time" function will appear"

III. Specific requirements

3.1 External interface requirements

User interfaces:

We use the Laravel framework to build this web application. And we use PHP for coding.

Hardware interfaces:

- Windows.

Software interface:

- Visual Studio code.

Communication interface:

- Discord
- Telegram

3.2 Functional requirements

In the following list, we have specified the functions that our app performs:

Functional Requirement	Function Requirements Description	Must/Want
FR 1	The system outputs the data to the project manager, which will contain developers who do not have a certain amount of working time.	Must

FR2	The system outputs data to the project manager, which will show the amount of time the developer has worked on the project for a certain amount of time to monitor the number of hours of work.	Must
FR3	The system makes an estimate of the remaining time of the specified project and outputs the information to the project manager.	Must
FR4	The manager can find out about the developer's reports and their details by the specified date that the system will issue.	Want
FR5	The system outputs data to the project manager, which will contain developers who have a certain amount of working time.	Must
FR6	The manager can find out about certain postponed activity times issued by the system.	Want
FR7	The system allows the project manager to see statistics that indicate how many developers have been working in a particular company for more than 5 years.	Want
FR8	The system can give the developer information about his last login.	Want
FR9	Employees can get information about the project and the people who worked on it and how much time was spent on this project.	Must
FR10	The system specifies the full name, activity, role of the work in the project, and the time spent on each employee's task who visited recently.	Must

FR11	The system can give information to the observer or the manager about the education of an employee.	Want
FR12	The system will give the manager information about the activities of a particular developer.	Must
FR13	The system will give the manager information about the status of each task of certain developers.	Must
FR14	The system allows the project manager to see statistics that indicate how many developers have successfully worked in the company.	Want
FR15	The system provides information about the level of the employee in a certain period of time and the project.	Must

3.3 System requirements

In general, on our web application "Active Monitoring", we will name the following systems as the main ones: time counting system, developer tracking system, daily reporting system, notification system. Let's describe each system separately

Time counting system

Status: Well done

With this system, the developer will be able to adjust their time, and the system will calculate the time that the developer spent on a specific task.

To control the time, the system will require the developer to select a specific activity, specify the time spent for the activity, and, if possible, comment on the actions. Next, the system calculates the time spent from the total time for this activity that the project manager has set and displays how much time developers have left for this activity. Also using this feature, the Project manager will have access to visibility such as: the status of each developer and efficiency. Also, if the time spent is 0, the project manager can assign another activity to the developer.

We will use the following data to calculate the time, since our web application is not yet fully ready, we provide you with a formula that calculates the time:

- $\text{total expected} = \text{hours spent} + \text{hours remaining}$

Current situation: We implemented this system taking into account the fact that the time spent and the status of developers is shown on the Dashboard page. The problem is that we previously had an incorrect time count, since the amount of time in the minus was issued. We took steps to fix it, but as it turned out, the problems were in the general data that we collected.

Developer tracking system

Status: Well done

With this system, the PM can view a general list of developers of their activities and time spent, the name of the developer in the Dashboard section. PM can also sort by developer status, and search for specific developer projects and view their activity. The PM can also view each developer's page(detailed information) separately in the Employees section.

To do this, the Project Manager must go to the Dashboard, Employees sections. To assign a new activity to a developer, the Project Manager must go to the Employees section and click on the "Assign project" button and sequentially specify the project, total time, developer name, and notification for the notification.

Current situation: We implemented this page to add the "Assign project" section to make it easier for the project manager to add activities for a free developer. The problem is that there are some shortcomings in terms of details, but we think to implement them in the future, such as: Select a project from the drop-down list, configure monitoring of those developers who have spent time for the project is equal to 0.

Notification system

Status: In Progress

With this system, the developer will receive notifications about the daily report. Track completed tasks every day, track performance, whether it's low (equals 1) or high (equals 5). In other words, our web application will ask the developer at the end of the day or at the end of the completed tasks, whether developer has completed everything, the time spent on this activity, and so on. Thus, at the beginning of the next day, our web application will be able to motivate the developer with this data for further work.

- **We will keep this private information with us**

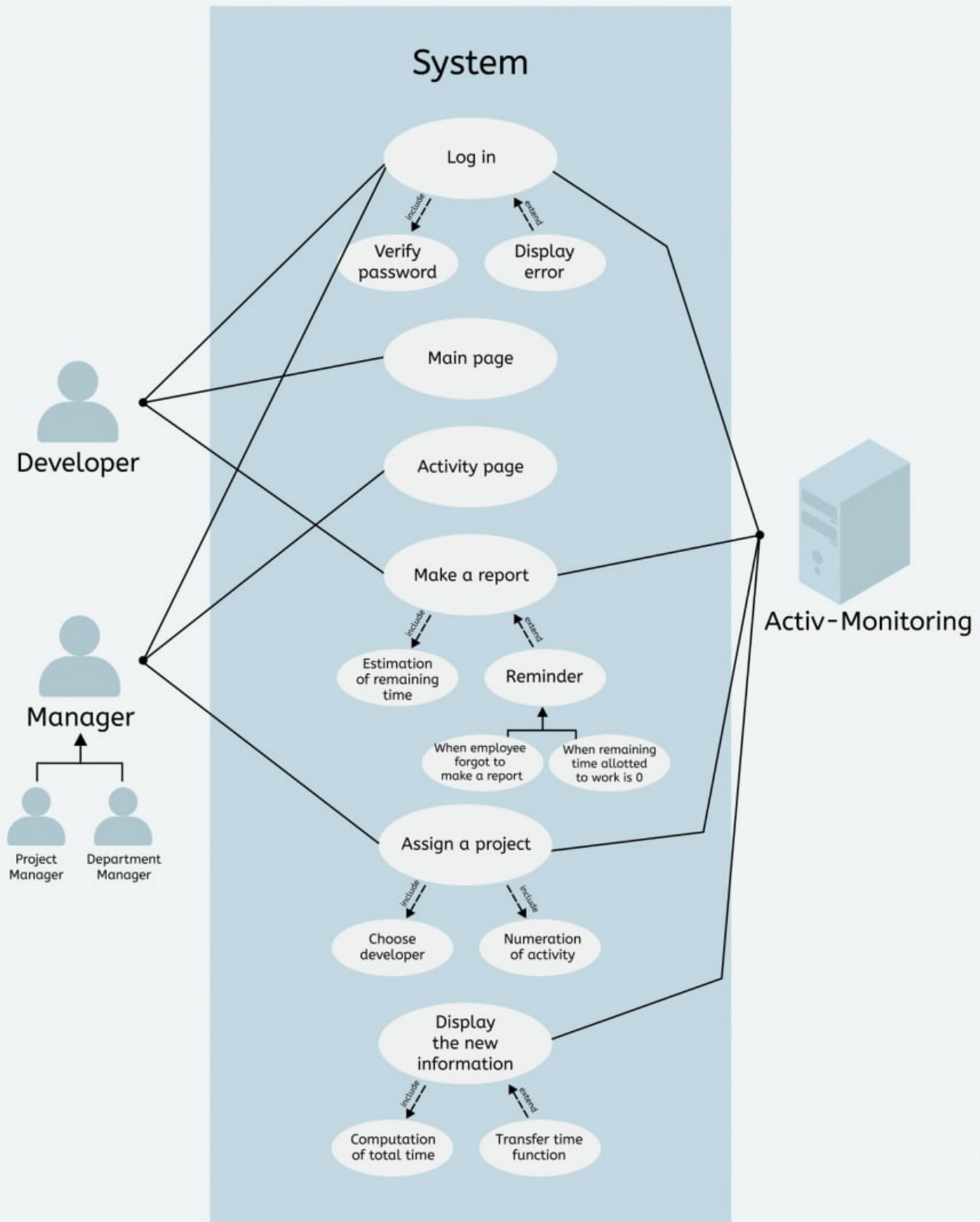
Current situation: During the creation process, it was a little hard to implement the notification system. For the structure, we thought about creating this logic: So that the Project Manager can specify what time notifications should arrive, because each company has a different working day and it is impossible to predict the total time for all.

Use case diagram

Link to Use Case Diagram in Figma:

<https://www.figma.com/file/VvT5TmTnCPgTEfqxvuc7N5/UML?node-id=0%3A1>

TripleA-UseCase-UML



IV. Process

4.1 Writing tables and databases

We met with the team to decide which tables we would use and what data we would collect. As it turned out, we changed the database structure several times during the project, as well as the data itself. These were the usual changes, since we didn't think about adding a table for users to our ERD to begin with. And then we decided to add such a table to save passwords and usernames. There were no difficulties in this case, but there were difficulties with collecting 5-6 thousand data, since we did not know how to write a ready-made script to read data from sites, and yet we learned to use such a library from python **Scrapy**, with which it was easy to collect data.

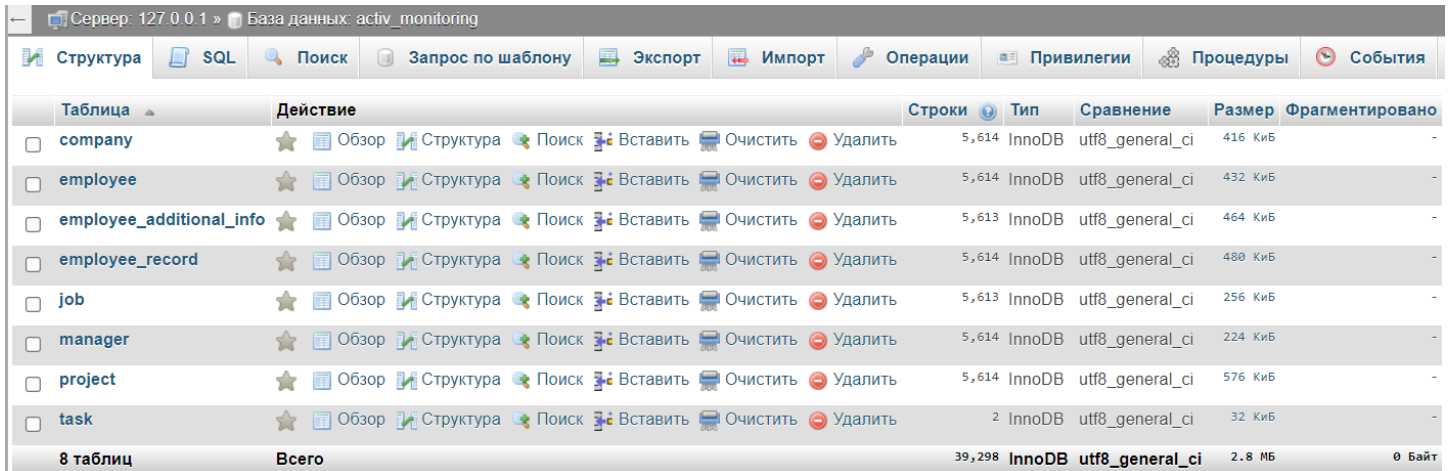


Таблица	Действие	Строки	Тип	Сравнение	Размер	Фрагментировано
company	★ Обзор Структура Поиск Вставить Очистить Удалить	5,614	InnoDB	utf8_general_ci	416 КиБ	-
employee	★ Обзор Структура Поиск Вставить Очистить Удалить	5,614	InnoDB	utf8_general_ci	432 КиБ	-
employee_additional_info	★ Обзор Структура Поиск Вставить Очистить Удалить	5,613	InnoDB	utf8_general_ci	464 КиБ	-
employee_record	★ Обзор Структура Поиск Вставить Очистить Удалить	5,614	InnoDB	utf8_general_ci	480 КиБ	-
job	★ Обзор Структура Поиск Вставить Очистить Удалить	5,613	InnoDB	utf8_general_ci	256 КиБ	-
manager	★ Обзор Структура Поиск Вставить Очистить Удалить	5,614	InnoDB	utf8_general_ci	224 КиБ	-
project	★ Обзор Структура Поиск Вставить Очистить Удалить	5,614	InnoDB	utf8_general_ci	576 КиБ	-
task	★ Обзор Структура Поиск Вставить Очистить Удалить	2	InnoDB	utf8_general_ci	32 КиБ	-
8 таблиц	Всего	39,298	InnoDB	utf8_general_ci	2.8 МБ	0 Байт

Img 1. Our tables before creating a database in the project

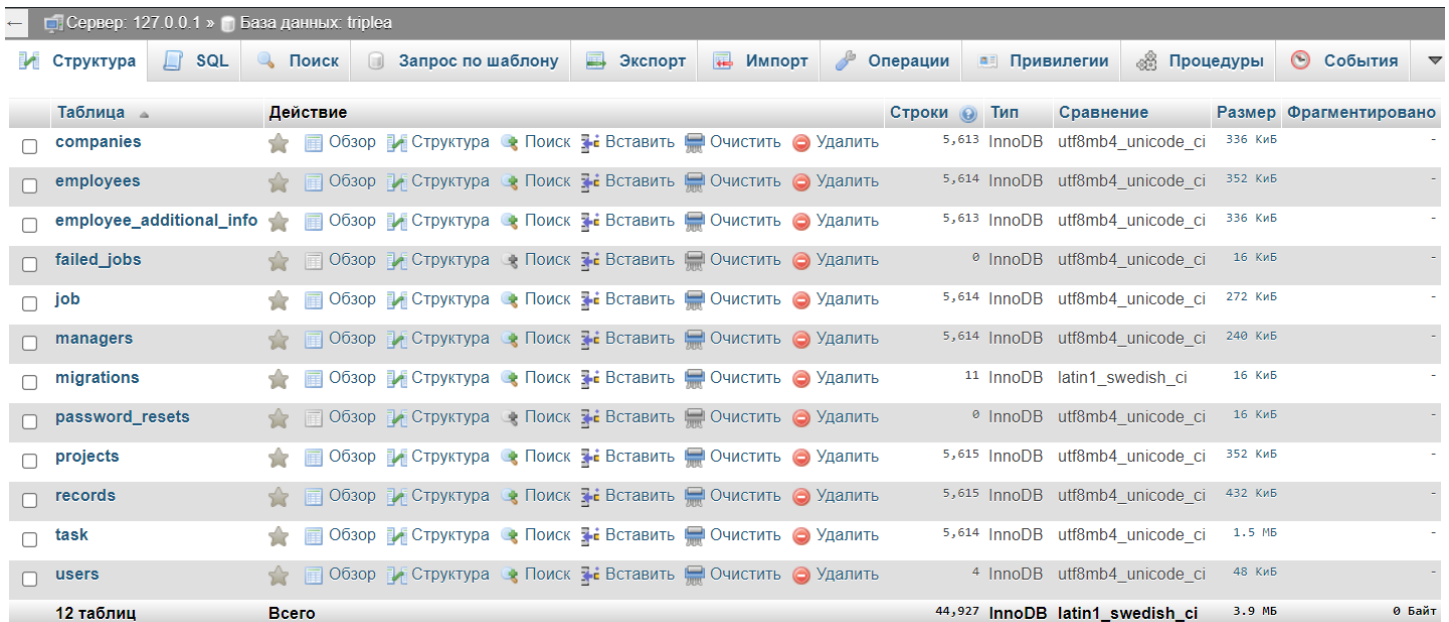


Таблица	Действие	Строки	Тип	Сравнение	Размер	Фрагментировано
companies	★ Обзор Структура Поиск Вставить Очистить Удалить	5,613	InnoDB	utf8mb4_unicode_ci	336 КиБ	-
employees	★ Обзор Структура Поиск Вставить Очистить Удалить	5,614	InnoDB	utf8mb4_unicode_ci	352 КиБ	-
employee_additional_info	★ Обзор Структура Поиск Вставить Очистить Удалить	5,613	InnoDB	utf8mb4_unicode_ci	336 КиБ	-
failed_jobs	★ Обзор Структура Поиск Вставить Очистить Удалить	0	InnoDB	utf8mb4_unicode_ci	16 КиБ	-
job	★ Обзор Структура Поиск Вставить Очистить Удалить	5,614	InnoDB	utf8mb4_unicode_ci	272 КиБ	-
managers	★ Обзор Структура Поиск Вставить Очистить Удалить	5,614	InnoDB	utf8mb4_unicode_ci	240 КиБ	-
migrations	★ Обзор Структура Поиск Вставить Очистить Удалить	11	InnoDB	latin1_swedish_ci	16 КиБ	-
password_resets	★ Обзор Структура Поиск Вставить Очистить Удалить	0	InnoDB	utf8mb4_unicode_ci	16 КиБ	-
projects	★ Обзор Структура Поиск Вставить Очистить Удалить	5,615	InnoDB	utf8mb4_unicode_ci	352 КиБ	-
records	★ Обзор Структура Поиск Вставить Очистить Удалить	5,615	InnoDB	utf8mb4_unicode_ci	432 КиБ	-
task	★ Обзор Структура Поиск Вставить Очистить Удалить	5,614	InnoDB	utf8mb4_unicode_ci	1.5 МБ	-
users	★ Обзор Структура Поиск Вставить Очистить Удалить	4	InnoDB	utf8mb4_unicode_ci	48 КиБ	-
12 таблиц	Всего	44,927	InnoDB	latin1_swedish_ci	3.9 МБ	0 Байт

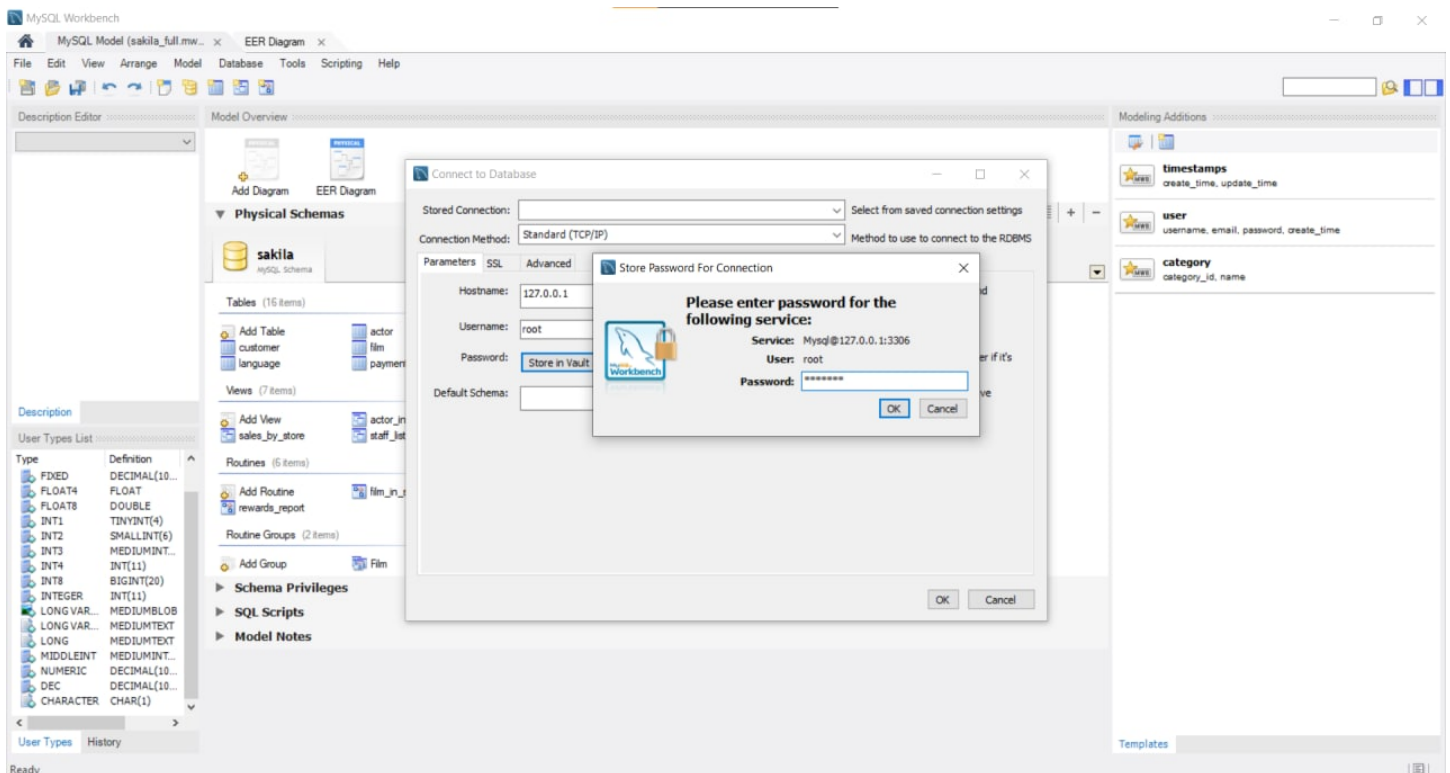
Img 2. Our tables after creating a database in the project

Сервер: 127.0.0.1 » База данных: triplea » Таблица: companies									
Обзор Структура SQL Поиск Вставить Экспорт Импорт Привилегии Операции Слежение Триггеры									
Структура таблицы Связи									
#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно	Действие	
1	id	bigint(20)	UNSIGNED	Нет	Нет	Нет	AUTO_INCREMENT	Изменить Удалить	Первичный Уникальный Индекс Ещё
2	employee_id	bigint(20)	UNSIGNED	Да	NULL	Нет	Нет	Изменить Удалить	Первичный Уникальный Индекс Ещё
3	name	varchar(191)	Да	NULL	Нет	Нет	Нет	Изменить Удалить	Первичный Уникальный Индекс Ещё
4	employee_number	varchar(191)	Да	NULL	Нет	Нет	Нет	Изменить Удалить	Первичный Уникальный Индекс Ещё
5	created_at	timestamp	Да	NULL	Нет	Нет	Нет	Изменить Удалить	Первичный Уникальный Индекс Ещё
6	updated_at	timestamp	Да	NULL	Нет	Нет	Нет	Изменить Удалить	Первичный Уникальный Индекс Ещё

Img 3. Attributes in the table companies after creating a database in the project

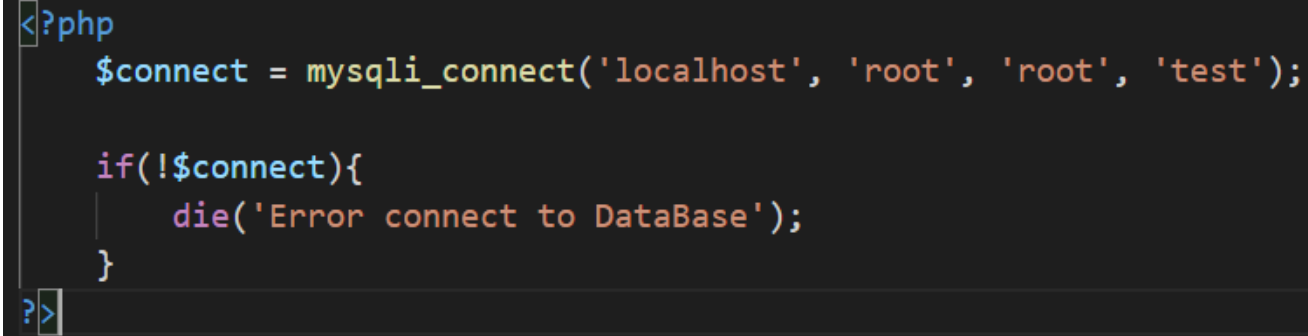
As you may have noticed, we added several tables that were not supposed to be there at first, since we did not know about the additional processes in the project implementation. The following fields were also added for each table: created_at, updated_at. The following changes were also made: the attribute names were changed to snake case, instead of camelCase.

4.2 Server side and connection



Img 4. Creating MySQL and connecting PHP

We tried to use the MySQL GUI for our project, but it took a lot of memory, and we considered it relevant to use this GUI despite the fact that there is a phpMyAdmin or MySQL extension on Visual Studio Code.

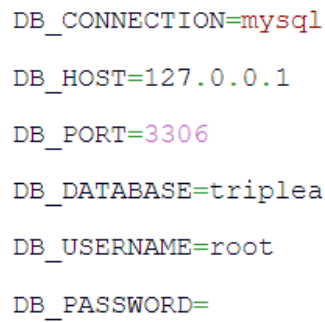
A screenshot of a code editor with a dark background. The code is written in PHP and shows a MySQL connection attempt. The code is as follows:

```
<?php
    $connect = mysqli_connect('localhost', 'root', 'root', 'test');

    if(!$connect){
        die('Error connect to DataBase');
    }
?>
```

Img 5. Connecting before using Laravel

There were no difficulties when connecting, as we used the following file .env when connecting:

A screenshot of a .env file with a light background. The file contains the following configuration for a MySQL database:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=triplea
DB_USERNAME=root
DB_PASSWORD=
```

Img 6. Connecting after using Laravel

To check if we successfully merged our arrival project into the database, we ran the migration command. The Laravel project by default comes with some default tables to store users and their password request. as soon as we ran the migrate command, it created the default tables in the database.

4.3 Laravel framework

Before starting the project, we faced such a problem that we could not properly link our pages, and using separate pages for PHP and then prescribing a separate service for each of them would be at least not logical. Therefore, the solution was to use Laravel, which made it much easier for us to implement. First, there are built-in migration files through which you could import data with just one command, and before that we used a script to add 5,000+ data to the tables. Secondly, it is convenient to use, while the code looks much cleaner than it was before (we had more than 1000 lines of code, which was not very logical). When implementing the project, we watched several video tutorials on how to use SQL queries to do manipulations with the data that we have. And since before that we had 2 courses related to the database, we could navigate how to roughly take data from the data database. With the help of our friend, we could write SQL queries and execute their functionality.

4.4 Frontend implementation

Since we have a small MVC project, we used a framework like Laravel. And in it to make the Front we used PHP. And the Blade template engine was also used. We chose Blade because it is a simple but powerful template engine that comes with Laravel. All Blade views are compiled into simple PHP code. The Blade view files use .blade.php file extension.

```
1 <?php
2
3 use Illuminate\Http\Request;
4 use Illuminate\Support\Facades\Route;
5
6 Route::middleware('auth:api')->get('/user', function (Request $request) {
7     return $request->user();
8 });
9
```

Since our front interacts with external resources that use our service, we first created api.php to use an external resource like Laravel. And all our resources that are used for the front are in the resources folder, and the display is in the views folder, and the styles are in css and js.

Let me show it:

```
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>Login</title>
5         <link rel="stylesheet" href="{{asset('css/main.css')}}">
6         <link href="{{asset('https://fonts.googleapis.com/css?family=Poppins:600&display=swap')}}" rel="stylesheet">
7         <script src="{{asset('https://kit.fontawesome.com/a81368914c.js')}}"></script>
8         <meta name="viewport" content="width=device-width, initial-scale=1">
9     </head>
10    <body>
11        @include('shared.nav')
12        @include('shared.header')
13        @yield('content')
14    </body>
15 </html>
```

```
<body>
    <div class="hero">
        <div class="form-box">
            <div class="button-box">
                <div id="btn"></div>
                <button type="button" class="toggle-btn" onclick="login()">Log In</button>
                <button type="button" class="toggle-btn" onclick="register()">Register</button>
            </div>
            <form id="login" class="input-group" method="post" action="{{route('signin')}}">
                {{ csrf_field() }}
                <input type="text" class="input-field" placeholder="User Name" required value="{{old('username')}}" name="username">
                <input type="password" class="input-field" placeholder="Enter Password" required value="{{old('password')}}" name="password">
                @if($errors->any())
                    <p class="text-danger">{{ $errors->first('error') }}</p>
                @endif
                <input type="checkbox" name="remember" class="check-box"><span>Remember Password</span>
                <button type="submit" class="submit-btn">Log In</button>
            </form>
            <form id="register" class="input-group" method="post" action="{{route('signup')}}">
                {{ csrf_field() }}
                <input type="text" class="input-field" placeholder="User Name" name="username">
                <input type="email" class="input-field" placeholder="e-mail" name="email">
                <input type="password" class="input-field" placeholder="Enter Password" name="password">
                <input type="checkbox" class="check-box" ><span>Agree</span>
                <button type="submit" class="submit-btn">Register</button>
            </form>
        </div>
    </div>
</body>
```

For login page


```

1 @extends('layout.main')
2 @section('content')
3     <div class="container">
4         <div class="filter">
5             <div></div>
6             <div style="display: flex">
7                 <div class="reload">
8                     
9                 </div>
10                <select class="minimal">
11                    <option selected value="today">{{ __('shared.today') }}</option>
12                    <option value="yesterday">{{ __('shared.yesterday') }}</option>
13                    <option value="lastseven">{{ __('shared.17d') }}</option>
14                    <option value="lastthird">{{ __('shared.130d') }}</option>
15                    <option value="thisweek">{{ __('shared.thisweek') }}</option>
16                    <option value="lastweek">{{ __('shared.lastweek') }}</option>
17                    <option value="thismonth">{{ __('shared.thismonth') }}</option>
18                    <option value="lastmonth">{{ __('shared.lastmonth') }}</option>
19                    <option value="thisyear">{{ __('shared.thisyear') }}</option>
20                    <option value="lastyear">{{ __('shared.lastyear') }}</option>
21                </select>
22                <select class="minimal" name="status">
23                    <option selected disabled>{{ __('shared.alluser') }}</option>
24                </select>
25            </div>
26        </div>
27        <div class="inner-wrap">
28            <div class="inner-wrap_container">
29                <div class="activities">Activities</div>
30                <div class="dashboard">
31                    @foreach($projects as $project)
32                        <div class="dashboard-card">
33                            <div class="box">
34                                <div>
35                                    <div class="name">{{ $project->name }}</div>
36                                    <div class="description">{{ $project->desc }}</div>
37                                </div>
38                                <div>
39                                    
40                                </div>
41                            </div>
42                            <div class="remaining-time">
43                                {{ $project->remaining_time }} {{ __('shared.hours') }}
44                            </div>
45                        </div>
46                    @endforeach
47                </div>
48            </div>
49        </div>
50    </div>
51 @endsection

```

First, of course, after login, we have a dashboard, which is like the main page.

```

1 @extends('layout.main')
2 @section('content')
3     <div class="container">
4         <div class="container--inner">
5             <div class="breadcrumb">
6                 <div class="main-page">{{ __('shared.employee') }}</div>
7                 
8                 <div class="username">{{ $result[0]->fullname }}</div>
9             </div>
10            <div class="profile">
11                
12                <div class="names">
13                    <div class="job">{{ $result[0]->job_role }}</div>
14                    <div class="username">{{ $result[0]->fullname }}</div>
15                    <button class="assign">
16                        
17                        {{ __('shared.assign') }}
18                    </button>
19                </div>
20            </div>
21            <div class="contact-info">
22                <div class="job">{{ __('shared.jobtype') }}: <p class="job-title">{{ $result[0]->job_role }}</p> </div>
23                <div class="job">{{ __('shared.workhours') }}: <p class="job-title">9:00 - 17:00</p> </div>
24                <div class="job">{{ __('shared.email') }}: <p class="job-title">shereism@mail.here</p> </div>
25                <div class="job">{{ __('shared.phone') }}: <p class="job-title">+7-777-77-77</p> </div>
26            </div>
27            <div class="breadcrumb">
28                <div class="main-page">{{ __('shared.projects') }}</div>
29            </div>
30            <div class="projects">
31                @foreach($projects as $project)
32                    <div class="post">
33                        
34                        <div class="name">{{ $project->name }}</div>
35                        <div class="description">{{ $project->desc }}</div>
36                    </div>
37                @endforeach
38            </div>
39        </div>
40    </div>
41 @endsection

```

Employees page

```

<div class="notify-box">
  <div class="error"></div>
  <div class="notify_header">
    <div class="notify_title">Attention Notification</div>
    <div class="time">23:59</div>
  </div>
  <div class="desc">Hello, Jake! You forgot to submit a ...</div>
</div>
<div class="notify-box">
  <div class="success-box"></div>
  <div class="notify_header">
    <div class="notify_title">Success Notification</div>
    <div class="time">23:59</div>
  </div>
  <div class="desc">Good Job, Jake! Your work sent to ...</div>
</div>
<div class="notify-box">
  <div class="warning"></div>
  <div class="notify_header">
    <div class="notify_title">Neutral Notification</div>
    <div class="time">23:59</div>
  </div>
  <div class="desc">Your work is in progress. Please, remember to ...</div>
</div>

```

Notifications

```

<div class="settings">
  <div class="lang">{{ __('shared.change_lang') }}:</div>
  <form method="post" action="{{route('localization', app()->getLocale())}}">
    {{ csrf_field() }}
    <select name="local" class="minimal" id="" onchange="this.form.submit()">
      <option value="en" @if(app()->getLocale()=='en') selected @endif>English</option>
      <option value="ru" @if(app()->getLocale()=='ru') selected @endif>Русский</option>
    </select>
  </form>
</div>

```

Settings

4.5 Backend implementation

There is a route in the project by which certain pages are displayed depending on the specified url. Routes respond to a method call from the controller. To display the page, the "index" function is used, which returns the generated page. Routes can be viewed in router / web.php

Controller - contains methods for interacting with the user (displaying, saving, deleting, editing, and so on).

Models - the main business logic is running. Working with the base, subtraction or something else.

For example: displaying the ru / employees page. When you follow this link, a method from the employees controller index is called through the route. The index method refers to the employees model, where your sql queries are written. The model returns the received data to the controller, the controller collects and displays the template we need.

In the index method, it refers to the employees model, where your sql queries are written in turn.

The model returns the received data to the controller, the controller collects it and displays the template we need.

4.7 Using SQL queries for functionality

For this part, we did not have any difficulties, while we used additional SQL queries to add, delete, update some of the data that is used in our project, for example, the following queries:

Img 7. Inserting data to the job table

```
class Job extends Model
{
    use HasFactory;
    public static function createJob($obj) {
        return DB::insert('insert into job (job_level, job_role) values (?, ?)', [1, $obj->job]);
    }
}
```

```
class Project extends Model
{
    use HasFactory;

    public static function createProject($obj, $id) {
        return DB::insert('insert into projects (employee_id, name) values (?, ?)', [$id, $obj->job]);
    }
}
```

Img 8. Inserting data to the project table

```
class Record extends Model
{
    use HasFactory;
    public static function createRecord($obj, $id) {
        $date = new DateTime(); //this returns the current date time
        return DB::insert('insert into records (employee_id, performance_rating, spent_time, standard_hours, total_w
        [$id, 3, '00:00:00', 8, 0, $date->format('Y-m-d-H-i-s'),$date->format('Y-m-d-H-i-s')]);
    }
}
```

Img 9. Inserting data to the records table

```
function createEmployee($name) {
    explode(' ', $name);
    DB::insert('insert into employees (first_name, last_name) values (?, ?)', [$arr[0], count($arr) > 1 ? $arr[1] : '']);
}
```

Img 10. Inserting data to the employee table

It was a bit difficult when implementing users: admin and regular user. After all, at the entrance of the admin, there are special sections that must be taken into account.

4.8 Optimizing SQL queries

We had a good experience with query optimization, because as we noticed, it takes a lot of time for the big data process, and you can get around this time by using optimization.

For example, we will analyze a 12-query SQL, namely

```
SELECT p.projectName, CONCAT(e.firstName, ' ', e.lastName) as "fullname", t.status AS "status"
FROM employee e, project p, task t, employee_record ea
WHERE e.employeeID = p.employeeID AND ea.employeeID = e.employeeID
AND ea.employeeRecordID = t.employeeRecordID;
```

Img 11. SQL for 12- query

And the time it takes to execute this request takes 4765.8228874206542969 ms. Now let's try to write our request under a different view, namely in the following form:

```
SELECT p.projectName, CONCAT(e.firstName, ' ', e.lastName) as "fullname", t.status AS "status"
FROM task t JOIN (employee_record ea JOIN employee e ON ea.employeeID = e.employeeID)
ON ea.employeeRecordID = t.employeeRecordID
JOIN project p ON e.employeeID = p.employeeID
```

Img 12. Rewritten SQL for 12- query ((employee_record * employee) * task * project)

```
RESULT: UglifyJS
RESULT: prepack
RESULT: prepack
RESULT: CodeIgniter
RESULT: CodeIgniter
RESULT: greenDAO
RESULT: greenDAO

TIME FOR testQuery: 9.591002225875854 seconds
+++++
+++++
Close database: triplea.sqlite
success
+++++
```

Img 12. Time optimization for 12-query

Rewriting the request resulted in a loss of time that amounted to 9591.0022258758544922 ms. This made it clear to us that under a certain type of spelling, a query can give both a positive and a negative outcome. As we could see the example above was about wasting time. But if we take the following spelling:

```
SELECT p.projectName, CONCAT(e.firstName, ' ', e.lastName) as "fullname", t.status AS "status"
FROM task t JOIN (project p JOIN employee e ON e.employeeID = p.employeeID
JOIN employee_record ea ON ea.employeeID = e.employeeID)
ON ea.employeeRecordID = t.employeeRecordID;
```

Img 13. Rewritten SQL for 12- query (project * employee * employee_record) * task

```
RESULT: UglifyJS
RESULT: prepack
RESULT: prepack
RESULT: CodeIgniter
RESULT: CodeIgniter
RESULT: greenDAO
RESULT: greenDAO

TIME FOR testQuery: 10.263847351074219 seconds

+++++
+++++
Close database: triplea.sqlite
success
+++++
```

Img 12. Time optimization for 12-query

Rewriting the request resulted in a loss of time that amounted to 10263.84735107421875 ms which also led to a loss of time.

V. Planning

5.1 Team Structure

Member	Position	Description
Aimzhan Sytdykova	UI/UX Designer, Frontend developer	Aimzhan worked on the design of the project, making up the logic between the sections in the app. Also make a Frontend.
Aruzhan Serikbayeva	Project Manager, Full-stack developer	Aruzhan put together a team and at the same time we were able to properly allocate responsibilities and work together. Also wrote code for the project.

Assylan Aitmambet	Backend developer	Assylan implemented the app. Also was good at teamwork and helped with implementation.
-------------------	-------------------	--

6 Conclusion

In the end, we want to note that the goal was completed correctly. It was a good experience for us to use such big data and use it to create an application that is relevant at the moment. It was also a good experience for us when working with SQL queries and optimizing them, as we did not know that sometimes for big data, not using queries correctly can take such a long time for the process. Since at first it took more than a minute and waiting for the server to download all this data was slightly not logical. If we go through the stages of the project implementation, we want to say that with the help of these stages of the project implementation, we saw what work we have done and are very proud of this work. We also note that our web application "Activ Monitoring" is very relevant today, since everyone is now sitting at home because of COVID-19 and working from home is not very effective for many people, and thus lowers the level of the company as a whole, because project customers will not be happy with the project execution time. Thus, with the help of our web application, not only developers go to the plus (increasing their efficiency growth), but also the companies in which these developers work (increasing work efficiency) and customers (without waiting for more time to order). That will be a plus for the country as a whole, because the level of business will grow, and the level of qualification of workers will be high.

References

For building app:

- <https://youtu.be/5fTFHAWWRV4>
- <https://youtu.be/HKJDLXsTr8A>
- <https://youtu.be/bW8OAD5Vzo8>

For design part and drawing diagrams:

- [Figma.com](https://www.figma.com)
- [Draw.io](https://draw.io)

For searching data:

- <https://scapy.readthedocs.io/en/latest/>
- <https://www.kaggle.com/>