SERVER SIDE INTERNET PROGRAMMING REPORT
P3K (PULU-PULU PRIMARY KEY)

Lecturer:

Williem

Class:

CIT - 4

Created by:

1. Adellya Putri Setyaningsih (001202400140)
2. Anjelin Ligiana Nainggolan (001202400202)
3. Nasywa Chonifahtun Fiqrihiyah (001202400178)

INFORMATICS

FACULTY OF COMPUTER SCIENCE

PRESIDENT UNIVERSITY

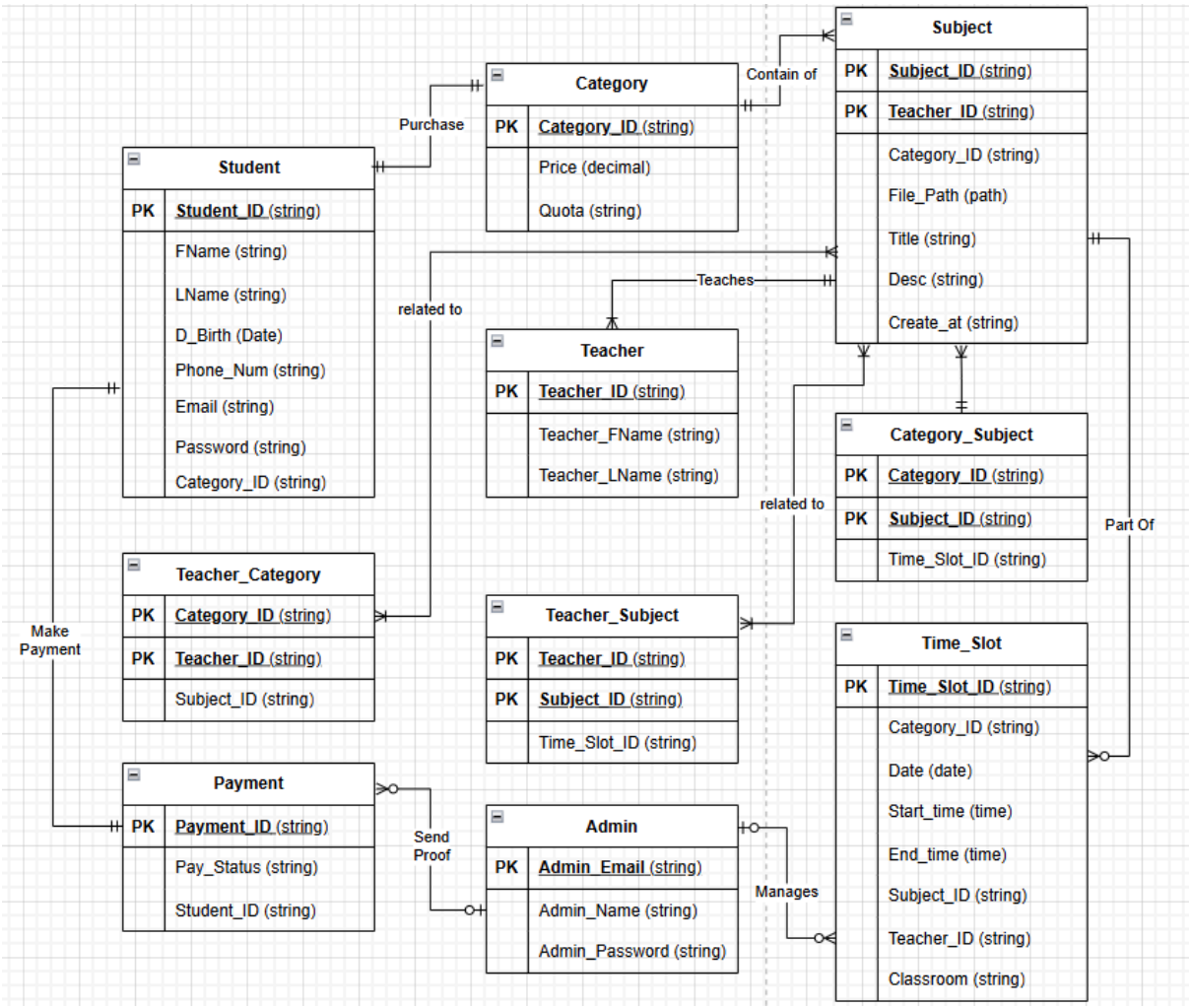
2024-2

TABLE OF CONTENTS

CHAPTER A.
SSIP PROJECT'S REQUIREMENTS

A.1. Tables Connected using Foreign Keys

This project uses 9 interconnected tables, each linked to others through foreign keys. The structure can be seen in Picture A.1.1.



Picture A.1.1 ERD Projects

In this Laravel project, we used migrations to create all the data structures, and we also generated dummy data using seeders. An example can be seen in Figure A.1.2 and Figure A.1.3.

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration {
    public function up(): void
    {
        Schema::create('subjects', function (Blueprint $table) {
            $table->string('subject_id'); // jangan pakai ->primary()
            $table->string('category_id')->nullable(); // Foreign key ke categories
            $table->string('file_path')->nullable();
            $table->string('title');
            $table->text('desc')->nullable();
            $table->timestamps(); // created_at & updated_at

            // Composite Primary Key
            $table->primary(['subject_id', 'title']);

            $table->foreign('category_id')->references('category_id')->on('categories')->onDelete('set null');
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('subjects');
    }
};
```

Figure A.1.2 Migrate Subject Table

```php
<?php

namespace Database\Seeders;

use App\Models\Subject;
use Illuminate\Database\Seeder;

class SubjectSeeder extends Seeder
{
    public function run()
    {
        Subject::create([
            'subject_id' => 'PPU',
            'category_id' => 'UTBK001',
            'file_path' => 'path/to/file.pdf',
            'title' => 'Matematika Dasar',
            'desc' => 'Materi dasar matematika untuk kelas awal',
        ]);

        // Contoh data lainnya
        Subject::create([
            'subject_id' => 'Fisika',
            'category_id' => 'SMA002',
            'file_path' => 'another/path/file.docx',
            'title' => 'Fisika Kelas 10',
            'desc' => 'Materi fisika untuk siswa kelas 10',
        ]);

        Subject::create([
            'subject_id' => 'Matematika',
            'category_id' => 'SMA002',
            'file_path' => 'another/path/file.docx',
            'title' => 'Matematika Kelas 10',
            'desc' => 'Materi fisika untuk siswa kelas 10',
        ]);
```

Figure A.1.3 Seeder Subject Table

A.2. CRUD Processes

The CRUD process is present in every part of the project, as it includes create, read, update, and delete operations. However, the complete CRUD implementation in this project can be found in several forms, specifically in the admin and student sections.

In the admin section, CRUD is implemented for almost all tables, since the admin controls all the inputs in this web course project.

- Admin: student table, payment table, category table, subject table, time slot table, teacher table, and admin table.
- Student: registration, edit profile.

A.2.1 Create

The Create process occurs during student registration, adding new subjects, categories, time slots, teachers, and entries in the Admin list. Figure A.2.1.1 shows the create form used for registration, while Figure A.2.1.2 illustrates the route configuration where the create action uses the "POST" method.

```
<!-- Formulir registrasi -->
<form action="{{ route('register.student') }}" method="POST">
  @csrf <!-- CSRF token Laravel -->

  {{-- First Name --}}
  <div class="mb-2">
    <label class="block text-yellow-500 mb-2">First Name</label>
    <input type="text" name="first_name" value="{{ old('first_name') }}" required
           class="w-full px-4 py-2 border border-gray-300 rounded-md focus:ring-2 focus:ring-blue-400"
           placeholder="Enter your first name">
  </div>

  {{-- Last Name --}}
  <div class="mb-2">
    <label class="block text-yellow-500 mb-2">Last Name</label>
    <input type="text" name="last_name" value="{{ old('last_name') }}"
           class="w-full px-4 py-2 border border-gray-300 rounded-md focus:ring-2 focus:ring-blue-400"
           placeholder="Enter your last name">
  </div>

  {{-- Date of Birth --}}
  <div class="mb-2">
    <label class="block text-yellow-500 mb-2">Date of Birth</label>
    <input type="date" name="dob" value="{{ old('dob') }}" required
           class="w-full px-4 py-2 border border-gray-300 rounded-md focus:ring-2 focus:ring-blue-400">
  </div>

  {{-- Phone Number --}}
  <div class="mb-2">
    <label class="block text-yellow-500 mb-2">Phone Number</label>
    <input type="tel" name="phone" value="{{ old('phone') }}" required
           class="w-full px-4 py-2 border border-gray-300 rounded-md focus:ring-2 focus:ring-blue-400"
           placeholder="Enter phone number">
```

Figure A.2.1.1 register blade

```
// Register
Route::get('/register', [AuthController::class, 'showStudentRegistrationForm'])->name('register.student');
Route::post('/register', [AuthController::class, 'registerStudent'])->name('register.student.submit');
```

Figure A.2.1.2 Register Route

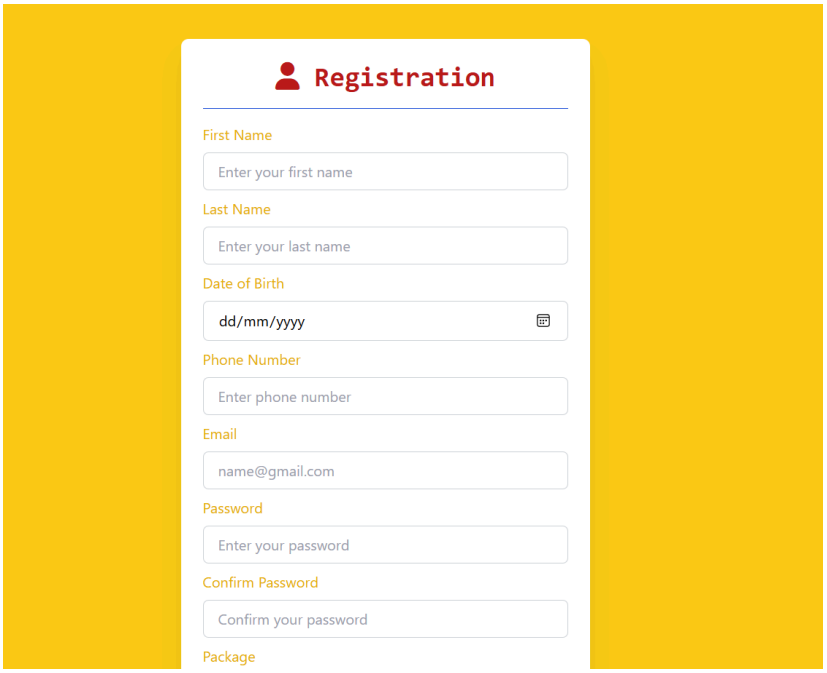


Figure A.2.1.3 register view

A.2.2 Read / Select

Read/Select refers only to the parts that retrieve or display data from the database.This is evidenced by the existence of Blade view files, as well as the index and show methods in the controller.

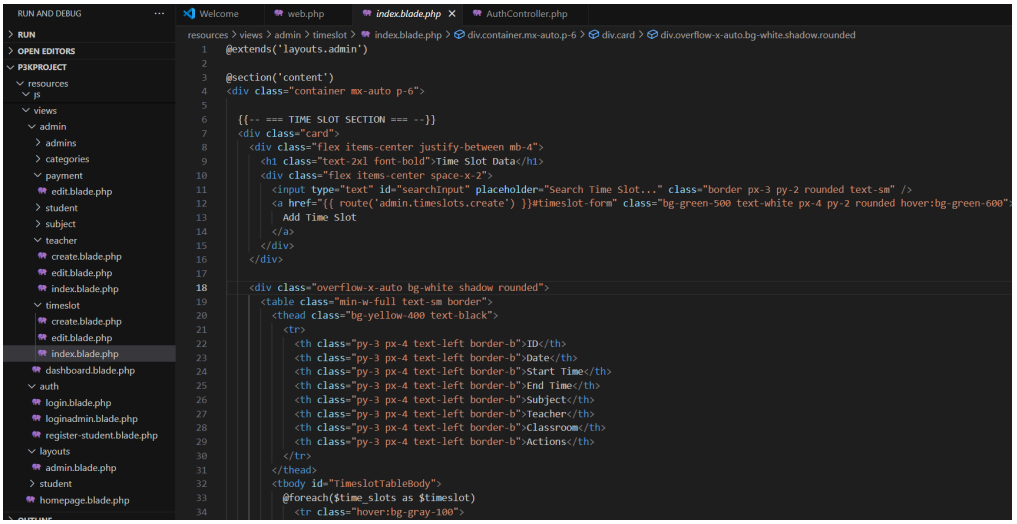


Figure A.2.2.1 index for time slot blade

A.2.3 Update

The Update process can be identified through the route using the "PUT" method, the update() method in the controller, the Edit form in the Blade view, and the connection to the database through the model.

```
// Route untuk edit kategori
Route::get('categories/{id}/edit', [CategorySubjectController::class, 'editCategory'])->name('categories.edit');
Route::put('categories/{id}', [CategorySubjectController::class, 'updateCategory'])->name('categories.update');

// Route untuk edit mata pelajaran
Route::get('subjects/{id}/edit', [CategorySubjectController::class, 'editSubject'])->name('subjects.edit');
Route::put('subjects/{id}', [CategorySubjectController::class, 'updateSubject'])->name('subjects.update');
```

Figure A.2.3.1 Router Edit and Update

```
@section('content')
<div class="container mx-auto p-6">
  <h1 class="text-2xl font-bold mb-6 text-center text-yellow-600">Edit Teacher</h1>

  <form action="{{ route('admin.teachers.update', $teacher->teacher_id) }}" method="POST" class="max-w-md mx-auto bg-white p-6 rounded shad
    @csrf
    @method('PUT')

    <div>
      <label class="block font-medium text-gray-700">Teacher ID</label>
      <input type="text" name="teacher_id" value="{{ $teacher->teacher_id }}" readonly class="w-full mt-1 p-2 border border-gray-300 bg-gray
    </div>

    <div>
      <label class="block font-medium text-gray-700">First Name</label>
      <input type="text" name="teacher_f_name" value="{{ $teacher->teacher_f_name }}" required class="w-full mt-1 p-2 border border-gray-30
    </div>

    <div>
      <label class="block font-medium text-gray-700">Last Name</label>
      <input type="text" name="teacher_l_name" value="{{ $teacher->teacher_l_name }}" required class="w-full mt-1 p-2 border border-gray-30
    </div>

    <div class="pt-4 flex justify-end space-x-2">
      <a href="{{ route('admin.teachers.index') }}"
         class="bg-blue-500 hover:bg-blue-600 text-white px-4 py-2 rounded">
        Back
      </a>

      <button type="submit"
              class="bg-yellow-500 hover:bg-yellow-600 text-white px-4 py-2 rounded">
        Update Teacher
      </button>
    </div>
```

Figure A.2.3.2 Blade for Edit and Update

A.2.4 Delete

To verify that the Delete process (data removal) is used in Laravel, you can refer to four main components: the route, the controller, the form (view), and the model.

```
// Route untuk delete kategori
Route::delete('categories/{id}', [CategorySubjectController::class, 'destroyCategory'])->name('categories.destroy');

// Route untuk delete mata pelajaran
Route::delete('subjects/{id}', [CategorySubjectController::class, 'destroySubject'])->name('subjects.destroy');
```

Figure A.2.4.1 Route Delete

```
            <form action="{{ route('admin.teachers.destroy', $teacher->teacher_id) }}" method="POST">
              @csrf
              @method('DELETE')
              <button type="submit" class="bg-red-500 text-white px-3 py-1 rounded ml-2">Delete</button>
            </form>
          </td>
        </tr>
```

Figure A.2.4.2 Form blade destroy/delete

CHAPTER B.
SSIP DESIGN CASE STUDY

B.1. Overview of the Case Study

This web project is called **P3K**, a web-based platform designed to support offline courses by providing an online system that stores learning materials based on the type or category of course registered by the user.

There are two main types of users in this system:

1. **Students:** can register and access learning materials according to the course category they have enrolled in.

2. **Admin:** has full access to post content, manage user data, and oversee all course-related information within the system.

The available courses focus specifically on categories such as **UTBK**, **SEKDIN**, and selected **high school (SMA)** subjects. The materials are organized by category to help students easily access the content relevant to their learning needs.

The main goal of this platform is to simplify the distribution of learning materials and enhance the efficiency of managing offline courses through a digital system, particularly in supporting students involved in academic preparation programs outside of school.

B.2. Business Logic

B.2.1 System Purpose

This system is designed to allow students to register for courses online. Once registered, students gain access to learning materials on the website, tailored to the specific course category they selected during registration.

In addition to study materials, the system also provides information such as the offline class schedule and the assigned teacher for each subject within the selected course. Therefore, this system supports both digital learning and the coordination of offline learning activities.

B.2.2 User Roles and Access Rights

The system has two main user roles: **Admin** and **Student**, each with the following access rights:

Admin

- Logs in using a predefined username and password.

- Has full access to perform CRUD operations on:
    - Student data (including editing student information and resetting passwords)
    - Teacher data
    - Course categories
    - Learning materials
    - Time slots/class schedules
- Verifies student payment records.
- Manages admin account credentials, including updating the username and password.

Student

- Registers through the registration form.
- Can log in only after being verified by the admin.
- Can view learning materials and class schedules based on the course category selected during registration.
- Can update their data and even profile picture through the student dashboard.

B.2.3 Workflow – Student & Admin

A. Student Workflow:

1. **HomePage (Website P3K, Promotion)**
2. **Registration**
   The student fills out the registration form → data saved to the students table.
3. **Manual Payment**
   The student contacts the admin via WhatsApp for payment.
4. **Verification by Admin**
   Admin verifies the payment and activates the student's account.
5. **Login & Dashboard Access**
   Once verified, the student can log in.
   The dashboard displays:
   - Learning materials (based on course category)
   - Class schedule (time slots)
   - Teacher information
6. **Profile Editing**
   Students can update their data, including profile picture.

B. Admin Workflow:

1. **Login**
   Admin logs in using a predefined username and password.
2. **Payment Verification**
   Admin verifies student payments and updates the status in the payments table.

3. **Data Management:**
   Admin performs **CRUD operations** on:

   - **Student Data** (students table):
     - Automatically created from registration
     - Can edit data (including password reset)
     - Can delete student records if needed
   - **Payment Data** (payments table):
     - Manually verified by admin
     - Marked as "verified" in the database
   - **Teacher Data** (teachers table):
     - Add, update, and delete teacher records linked to each course category
   - **Course Categories** (categories table):
     - Manage UTBK, SEKDIN, or SMA course categories
   - **Subjects** (subjects table):
     - Add/edit/delete course materials
     - Materials are displayed on student dashboards based on the selected course
   - **Time Slots** (time_slot table):
     - Define class schedules for each category
   - **Admin Account** (admins table):
     - Admin can change their own username and password

All data managed by the admin is displayed on the student dashboard, including materials, teacher names, and time slots.

B.2.4 Laravel Model Structure

1. **Admin Model**

   The **Admin** model uses the admins table with admin_email as its primary key, which is a non-incrementing string. It is responsible for handling admin authentication and access control, allowing admins to manage core data such as students, teachers, categories, subjects, payments, and schedules. While the model does not define direct Eloquent relationships, it plays a central role in system operations and data administration.

2. **Category Model**

   The Category model uses the categories table with category_id as its non-incrementing primary key. It is used to represent different course types such as UTBK, SEKDIN, and SMA, along with their associated price and quota. This model defines a one-to-many relationship with both Student and Subject, indicating that each category can have multiple students and multiple subjects assigned to it.

3. **CategorySubject Model**

   The CategorySubject model uses the category_subject table as a pivot table to manage the many-to-many relationship between Category and Subject. It contains the composite keys category_id and subject_id, which link records from both tables. This model does not define explicit Eloquent relationships but serves as the intermediary that connects categories to their respective subjects. Timestamps are disabled, and the primary key is non-incrementing due to the composite nature of the keys.

4. **Payment Model**

   The Payment model uses the payments table with pay_ID as a non-incrementing string primary key. It stores student payment transaction data, including the student_id and payment status, which defaults to 'pending'. This model defines a belongsTo relationship to the Student model via the student_id foreign key, linking each payment to the corresponding student who made the transaction.

5. **Students Model**

   The Student model uses the students table with student_id as its non-incrementing string primary key. It stores student personal data including first name, last name, date of birth, phone number, email, password, and the associated category_id. The model extends Laravel's Authenticatable to enable student authentication. It defines a belongsTo relationship with the Category model and a hasMany relationship with the Payment model, representing the student's course category and payment transactions, respectively. Sensitive fields like password and remember token are hidden, and email verification timestamps are cast to datetime.

6. **Subjects Model**

   The Subject model uses the subjects table with subject_id as a non-incrementing string primary key. It stores learning materials associated with a specific category, including fields such as file path, title, and description. The model defines a belongsTo relationship to the Category model, a many-to-many (belongsToMany) relationship with the Teacher model through the pivot table teacher_subject (which also stores time_slot_id), and a hasMany relationship with the TimeSlot model representing scheduled offline classes for the subject.

7. **Teachers Model**

   The Teacher model uses the teachers table with teacher_id as a non-incrementing string primary key. It stores teacher information including first name, last name, and classroom assignment. The model defines many-to-many (belongsToMany) relationships with the Category model via the pivot table teacher_category (which

includes a pivot field subject_id), and with the Subject model via the teacher_subject pivot table (which includes time_slot_id). Additionally, it has a one-to-many (hasMany) relationship with the TimeSlot model, representing the offline class schedules assigned to the teacher.

8. **Teachercategory Model**

The TeacherCategory model uses the teacher_category table as a pivot table to manage the many-to-many relationship between Teacher and Category, with an additional subject_id field to specify the subject taught by the teacher within that category. It has composite primary keys and disables timestamps. This model acts as an intermediary without explicit Eloquent relationship methods, connecting teachers to their assigned categories and subjects.

9. **Teachersubject Model**

The TeacherSubject model uses the teacher_subject table as a pivot table to manage the many-to-many relationship between Teacher and Subject, with an additional time_slot_id field to link the subject taught by the teacher to a specific class schedule. It has composite primary keys and disables timestamps. This model functions as an intermediary table without explicit Eloquent relationship methods.

10. **TimeSlot Model**

The TimeSlot model uses the time_slots table with time_slot_id as a non-incrementing string primary key. It stores scheduled offline class details, including date, start and end times, classroom, and associations to a specific category, subject, and teacher. The model defines belongsTo relationships with the Category, Subject, and Teacher models via their respective foreign keys, linking each timeslot to its course category, learning material, and instructor.

```php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;

class Admin extends Authenticatable
{
    use HasFactory;

    protected $table = 'admins'; // Pastikan nama tabel sesuai, sudah benar
    protected $primaryKey = 'admin_email';
    public $incrementing = false;
    protected $keyType = 'string';

    protected $fillable = [
        'admin_name',
        'admin_email',
        'admin_password',
    ];

    protected $hidden = [
        'admin_password',
        'remember_token',
    ];

    // Kasih tahu Laravel kalau password-nya bukan 'password' biasa
    public function getAuthPassword()
    {
        return $this->admin_password;
    }

    protected $casts = [
        'email_verified_at' => 'datetime',
    ];
}
```

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class Payment extends Model
{
    use HasFactory;

    protected $table = 'payments';

    protected $primaryKey = 'pay_ID'; // Tentukan nama kolom primary key
    public $incrementing = false;     // Jika pay_ID bukan auto-increment
    protected $keyType = 'string';    // Jika pay_ID bertipe string

    protected $fillable = [
        'student_id',
        'pay_status',
    ];

    // Set default status ke 'pending' saat membuat data
    protected $attributes = [
        'pay_status' => 'pending', // Default status
    ];

    /**
     * Relasi ke model Student
     */
    public function student(): BelongsTo
    {
        return $this->belongsTo(Student::class, 'student_id', 'student_id');
    }
}
```

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasMany;
use Illuminate\Database\Eloquent\Relations\BelongsToMany;

class Category extends Model
{
    use HasFactory;

    protected $primaryKey = 'category_id';
    public $incrementing = false;
    protected $fillable = [
        'category_id', 'price', 'quota'
    ];

    public function students() {
        return $this->hasMany(Student::class, 'category_id');
    }

    public function subjects() {
        return $this->hasMany(Subject::class, 'category_id');
    }

}
```

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class CategorySubject extends Model
{
    use HasFactory;

    protected $table = 'category_subject';
    protected $fillable = [
        'category_id',
        'subject_id',
    ];

    public $timestamps = false;
    public $incrementing = false; // Karena primary key komposit
}
```

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Student extends Authenticatable
{
    use HasFactory;

    protected $primaryKey = 'student_id';
    public $incrementing = false;
    protected $keyType = 'string';

    protected $fillable = [
        'student_id',
        'f_name',
        'l_name',
        'phone_num',
        'd_birth',
        'email',
        'password',
        'category_id',
    ];

    protected $hidden = [
        'password',
        'remember_token',
    ];

    protected $casts = [
        'email_verified_at' => 'datetime',
    ];
```

```php
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class TimeSlot extends Model
{
    use HasFactory;
    protected $primaryKey = 'time_slot_id';
    public $incrementing = false;
    protected $keyType = 'string';

    protected $fillable = [
        'time_slot_id',
        'category_id',
        'date',
        'start_time',
        'end_time',
        'subject_id',
        'teacher_id',
        'classroom',
    ];

    public function category(): BelongsTo
    {
        return $this->belongsTo(Category::class);
    }

    public function subject(): BelongsTo
    {
        return $this->belongsTo(Subject::class, 'subject_id', 'subject_id');
    }

    public function teacher(): BelongsTo
    {
        return $this->belongsTo(Teacher::class, 'teacher_id', 'teacher_id');
    }
```

```php
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsToMany;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Teacher extends Model
{
    use HasFactory;

    protected $primaryKey = 'teacher_id';
    public $incrementing = false;
    protected $keyType = 'string';

    protected $fillable = [
        'teacher_id',
        'teacher_f_name', // Sesuaikan dengan nama kolom di database
        'teacher_l_name', // Sesuaikan dengan nama kolom di database
        'classroom',
    ];

    public function categories(): BelongsToMany
    {
        return $this->belongsToMany(Category::class, 'teacher_category', 'teacher_id', 'category_id')->withPivot('subject_id');
    }

    public function subjects(): BelongsToMany
    {
        return $this->belongsToMany(Subject::class, 'teacher_subject', 'teacher_id', 'subject_id')->withPivot('time_slot_id');
    }

    public function timeSlots(): HasMany
    {
        return $this->hasMany(TimeSlot::class, 'teacher_id'); // Tambahkan foreign key

    }
}
```

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class TeacherCategory extends Model
{
    use HasFactory;

    protected $table = 'teacher_category';
    protected $fillable = [
        'teacher_id',
        'category_id',
        'subject_id',
    ];

    public $timestamps = false;
    public $incrementing = false; // Penting untuk primary key komposit
}
```

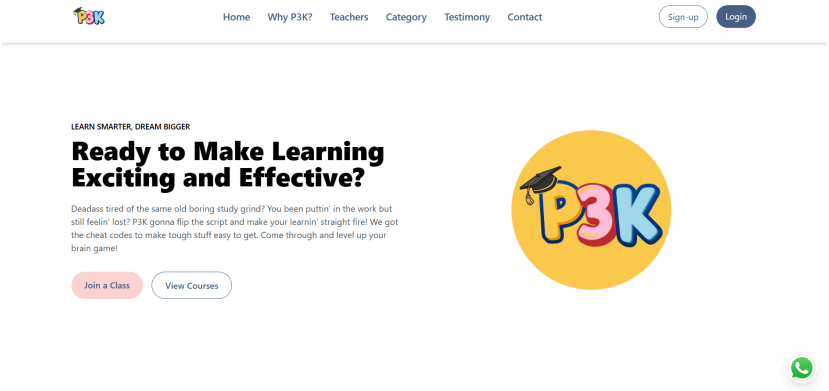
```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class TeacherSubject extends Model
{
    use HasFactory;

    protected $table = 'teacher_subject';
    protected $fillable = [
        'teacher_id',
        'subject_id',
        'time_slot_id',
    ];

    public $timestamps = false;
    public $incrementing = false; // Karena primary key komposit (teacher_id, subject_id)
}
```

```php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\BelongsToMany;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Subject extends Model
{
    use HasFactory;
    protected $table = 'subjects';
    protected $primaryKey = 'subject_id';
    public $incrementing = false;
    protected $keyType = 'string';
    protected $fillable = [
        'subject_id',
        'category_id',
        'file_path',
        'title',
        'desc',
    ];

    public function category(): BelongsTo
    {
        return $this->belongsTo(Category::class);
    }

    public function teachers(): BelongsToMany
    {
        return $this->belongsToMany(Teacher::class, 'teacher_subject', 'subject_id', 'teacher_id')->withPivot('time_slot_id');
    }

    public function timeSlots(): HasMany
    {
        return $this->hasMany(TimeSlot::class, 'subject_id', 'subject_id');
    }
```

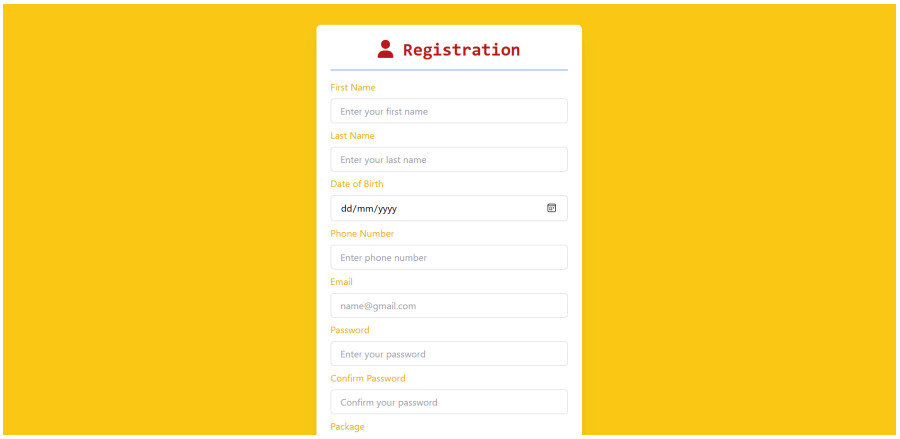
CHAPTER C.
IMPLEMENTATION OF PHP-BASED DATABASE APPLICATION

C.1. Overall Website

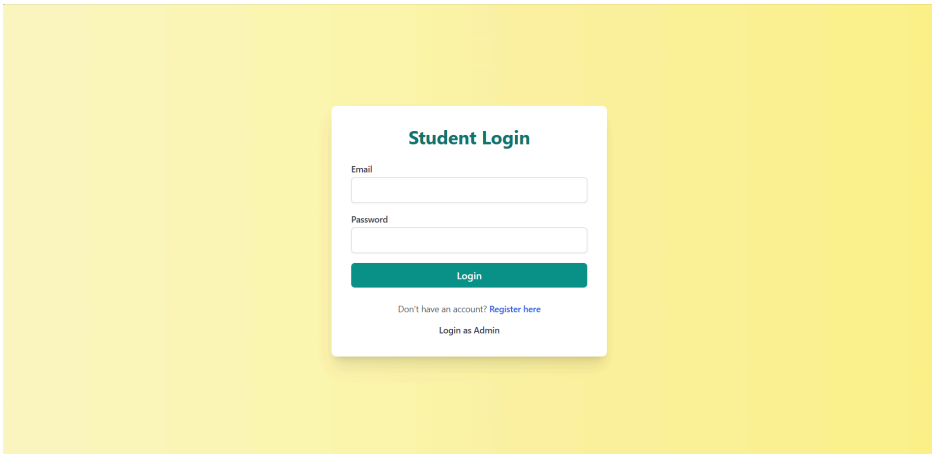
1. Homepage



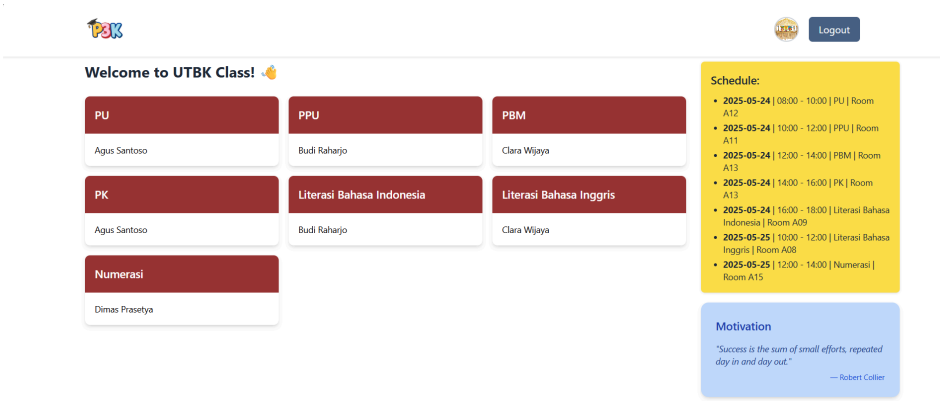
2. Register



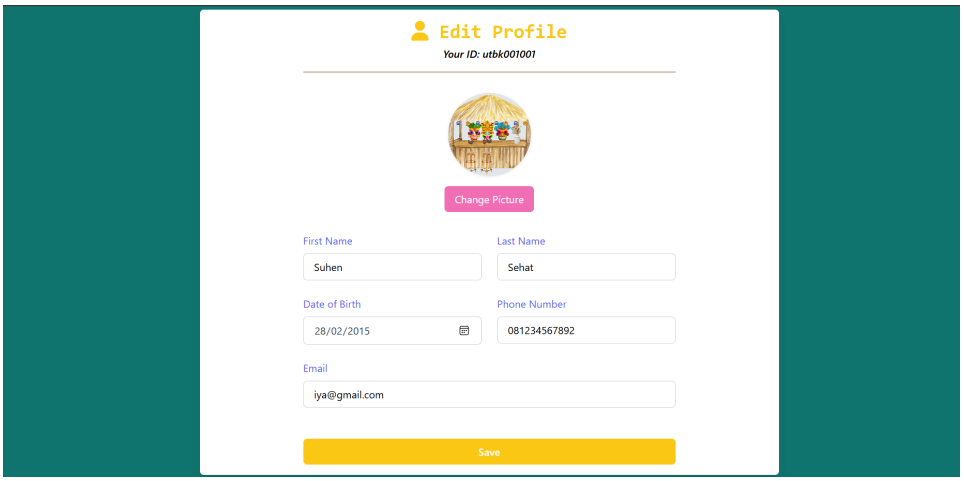
3. Login Student



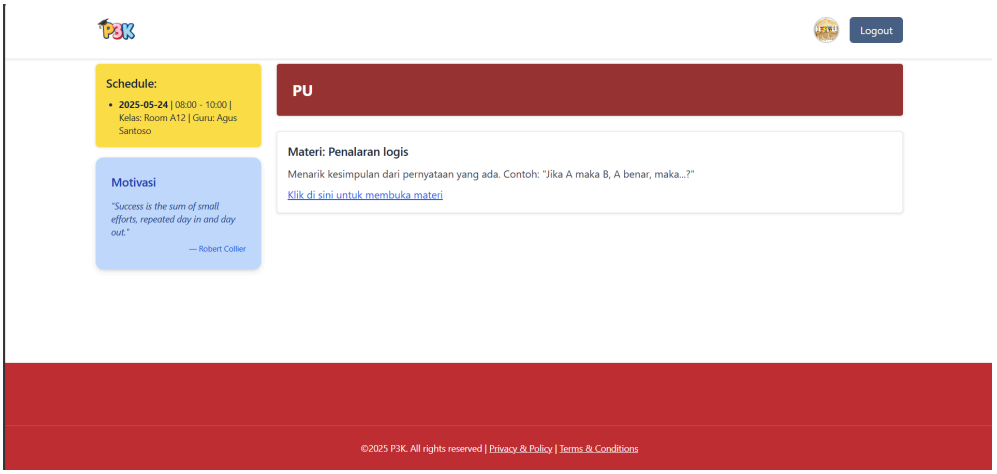
4. Dashboard Student



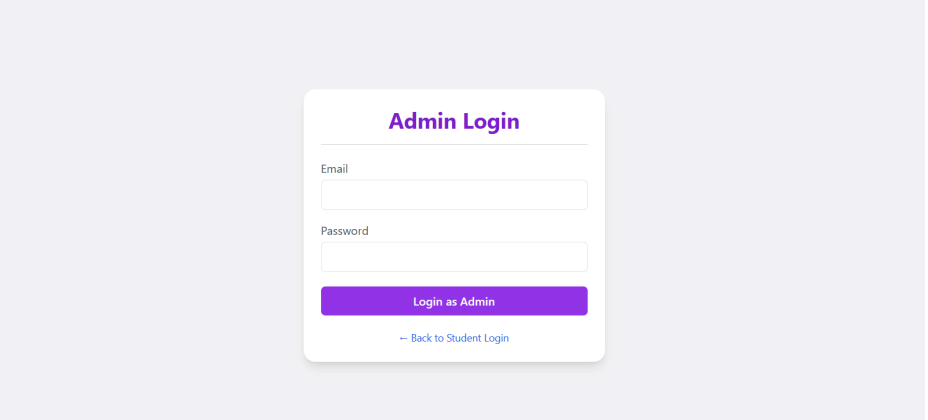
5. Edit Profile



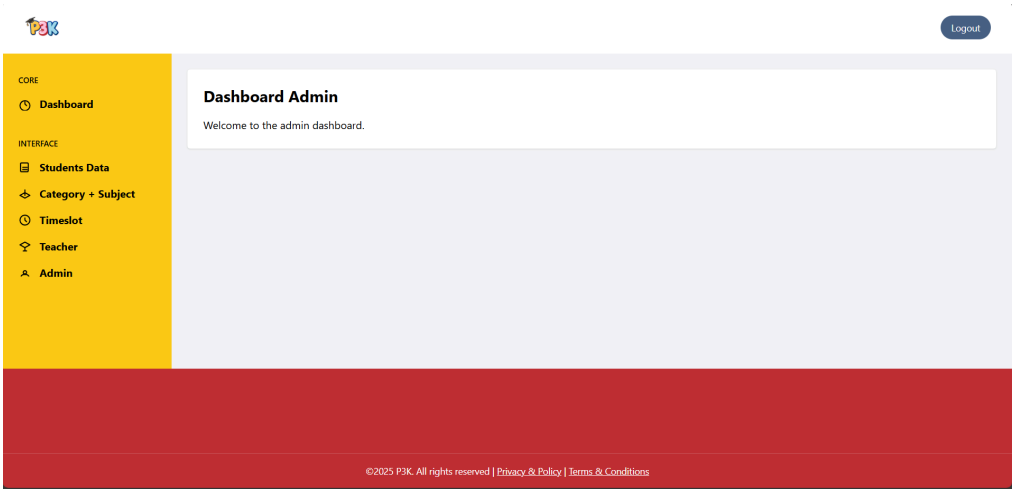
6. Student Material



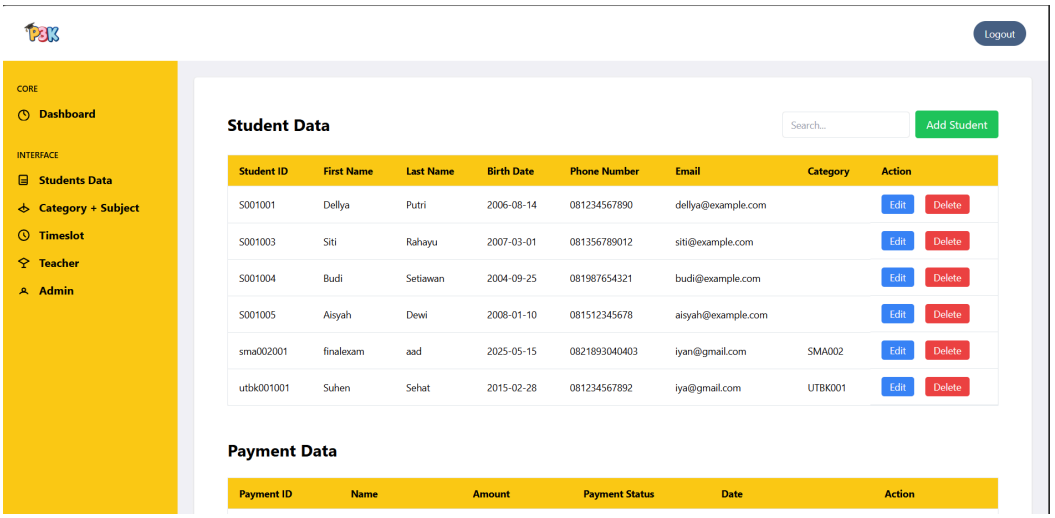
7. Login Admin

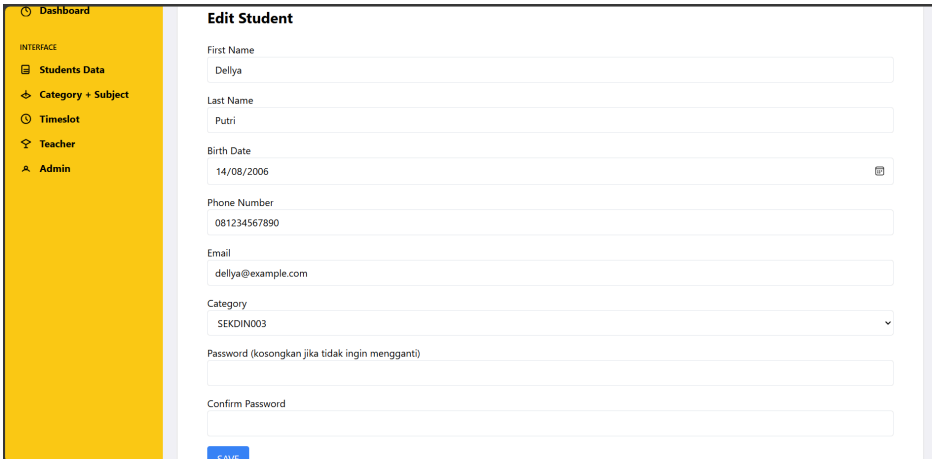


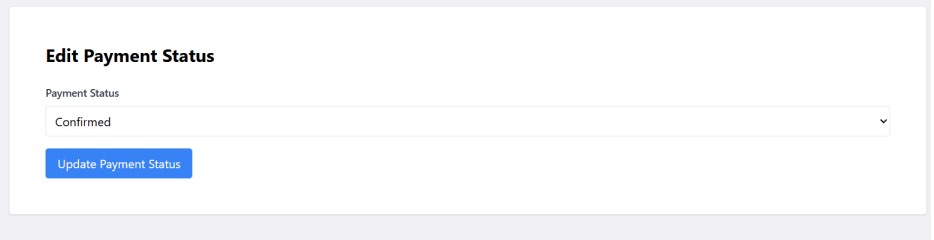
8. Dashboard Admin



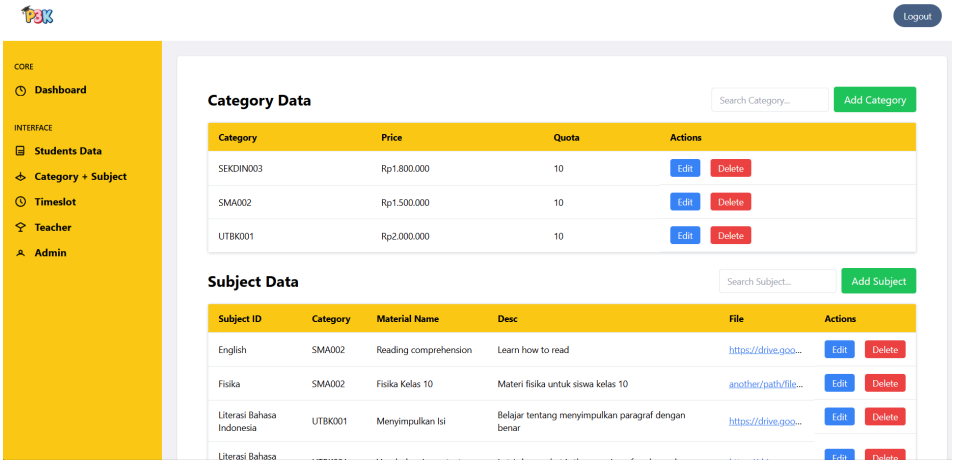
9. Student Data and Payment

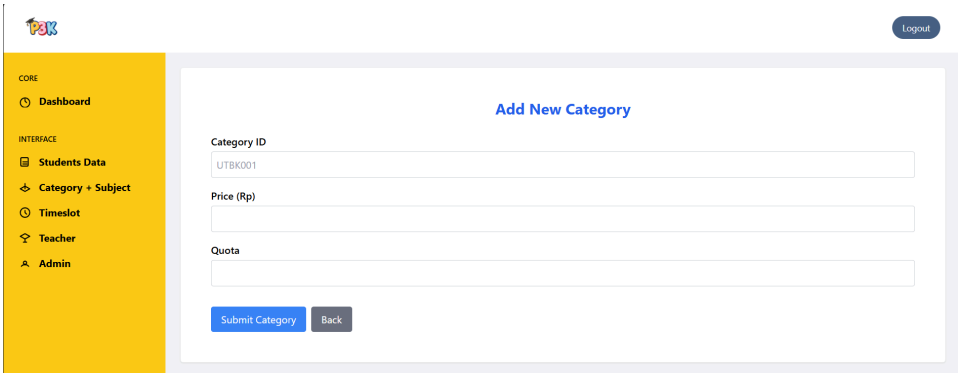


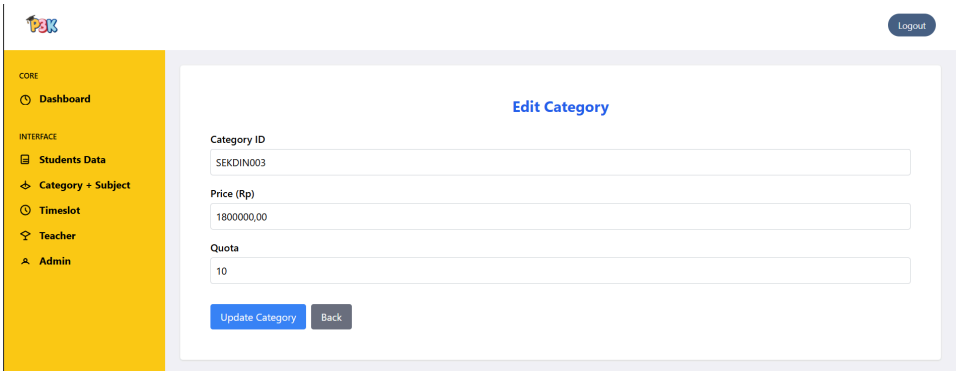


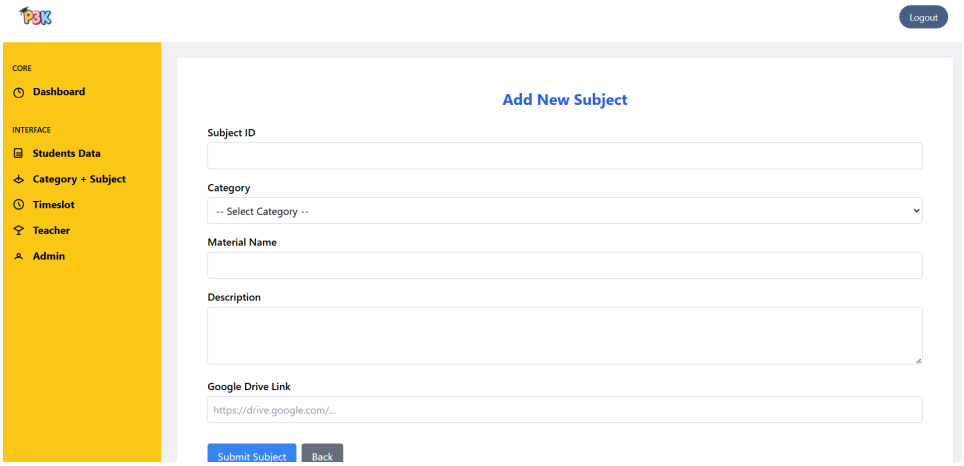


10. Category and Subject

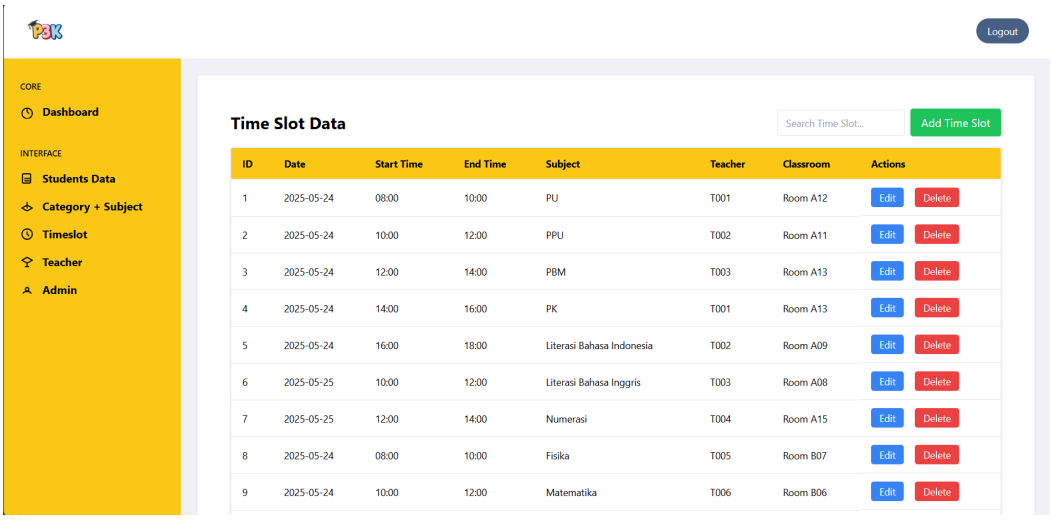


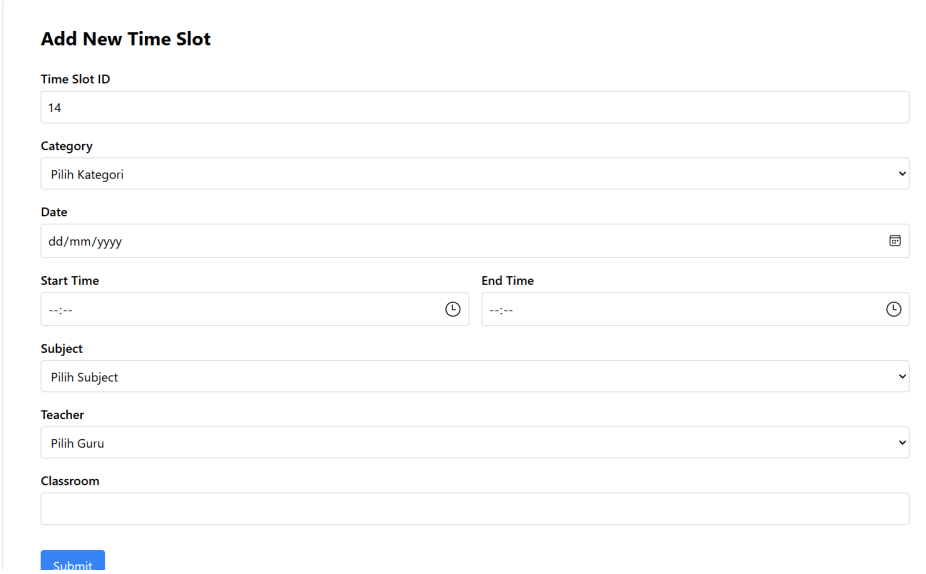


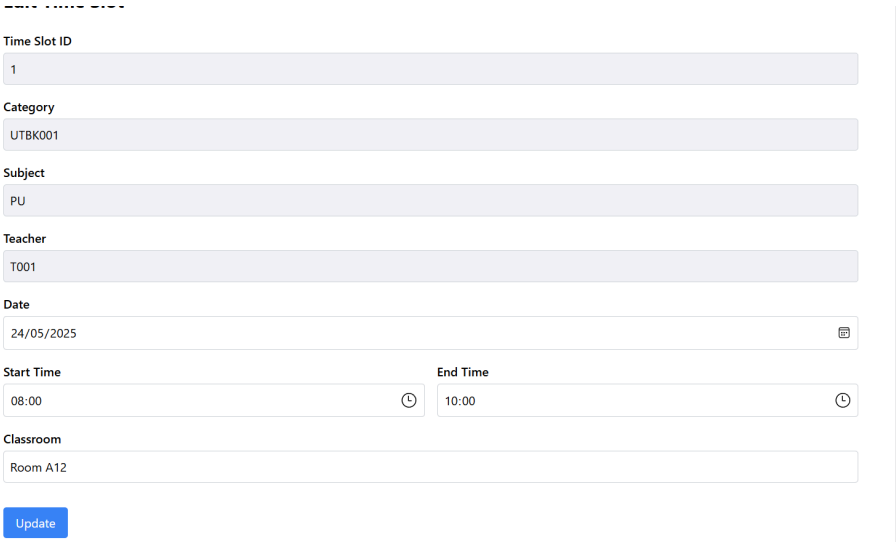

Logout
CORE
Dashboard
INTERFACE
Students Data
Category + Subject
Timeslot
Teacher
Admin
Edit Category
Category ID
SEKDIN003
Price (Rp)
1800000,00
Quota
10
Update Category
Back


Logout
CORE
Dashboard
INTERFACE
Students Data
Category + Subject
Timeslot
Teacher
Admin
Add New Subject
Subject ID
Category
-- Select Category --
Material Name
Description
Google Drive Link
https://drive.google.com/...
Submit Subject
Back

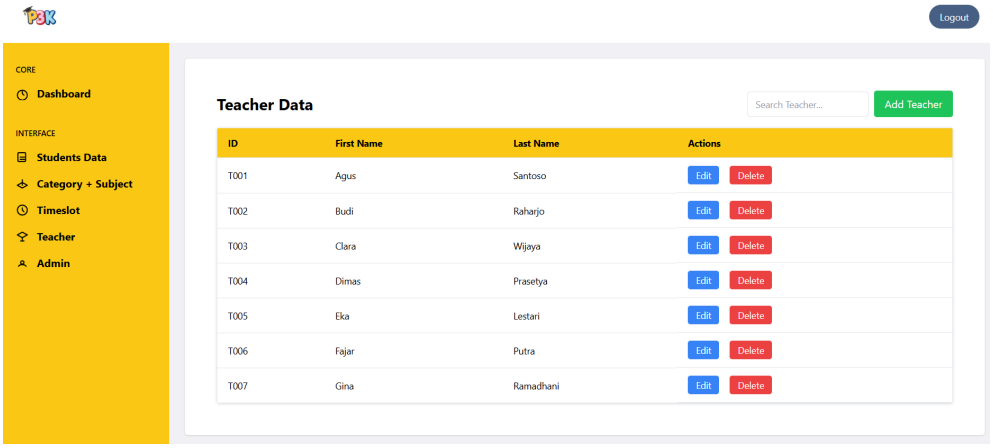
11. Time Slot

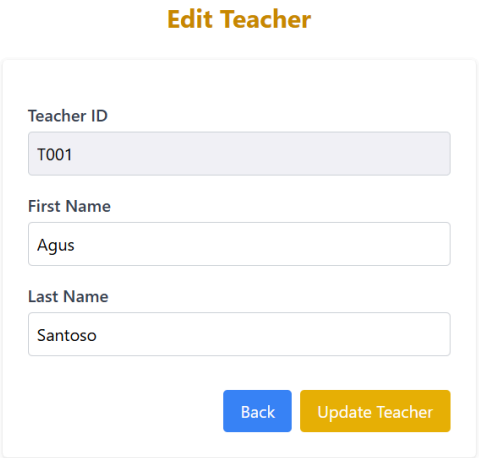

Logout
CORE
Dashboard
INTERFACE
Students Data
Category + Subject
Timeslot
Teacher
Admin
Time Slot Data
Search Time Slot...
Add Time Slot
ID Date Start Time End Time Subject Teacher Classroom Actions
1 2025-05-24 08:00 10:00 PU T001 Room A12 Edit Delete
2 2025-05-24 10:00 12:00 PPU T002 Room A11 Edit Delete
3 2025-05-24 12:00 14:00 PBM T003 Room A13 Edit Delete
4 2025-05-24 14:00 16:00 PK T001 Room A13 Edit Delete
5 2025-05-24 16:00 18:00 Literasi Bahasa Indonesia T002 Room A09 Edit Delete
6 2025-05-25 10:00 12:00 Literasi Bahasa Inggris T003 Room A08 Edit Delete
7 2025-05-25 12:00 14:00 Numerasi T004 Room A15 Edit Delete
8 2025-05-24 08:00 10:00 Fisika T005 Room B07 Edit Delete
9 2025-05-24 10:00 12:00 Matematika T006 Room B06 Edit Delete

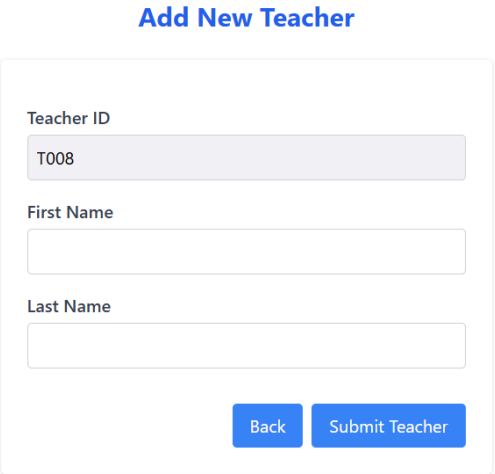

Add New Time Slot
Time Slot ID
14
Category
Pilih Kategori
Date
dd/mm/yyyy
Start Time
--:--
End Time
--:--
Subject
Pilih Subject
Teacher
Pilih Guru
Classroom
Submit


Time Slot ID
1
Category
UTBK001
Subject
PU
Teacher
T001
Date
24/05/2025
Start Time
08:00
End Time
10:00
Classroom
Room A12
Update

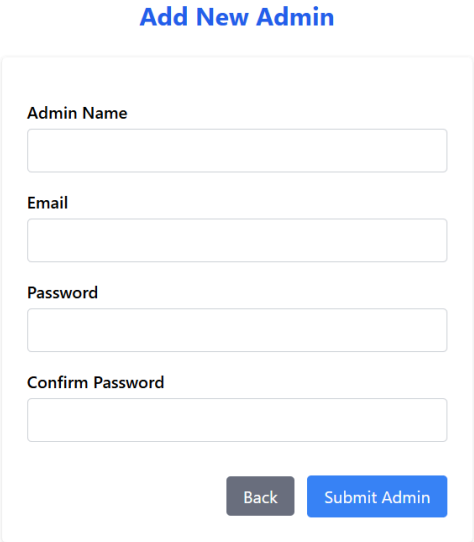
12. Teacher

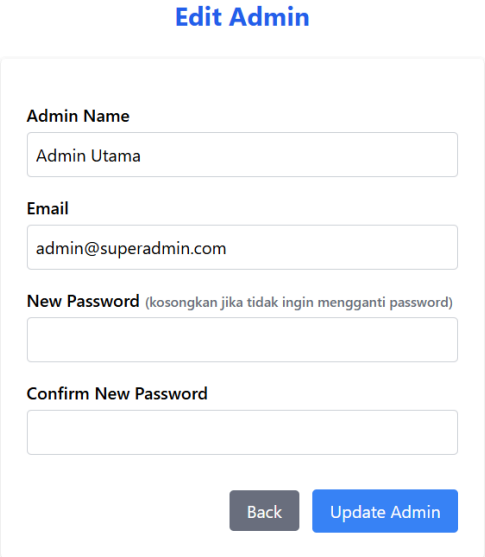

Logout
CORE
Dashboard
INTERFACE
Students Data
Category + Subject
Timeslot
Teacher
Admin
Teacher Data
Search Teacher...
Add Teacher
ID
First Name
Last Name
Actions
T001 Agus Santoso Edit Delete
T002 Budi Raharjo Edit Delete
T003 Clara Wijaya Edit Delete
T004 Dimas Prasetya Edit Delete
T005 Eka Lestari Edit Delete
T006 Fajar Putra Edit Delete
T007 Gina Ramadhani Edit Delete

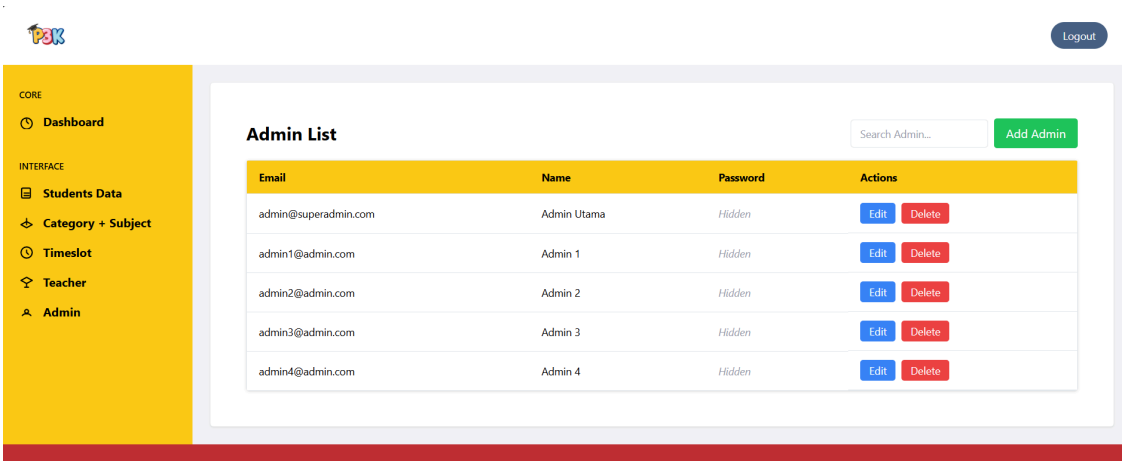

Edit Teacher
Teacher ID
T001
First Name
Agus
Last Name
Santoso
Back
Update Teacher


Add New Teacher
Teacher ID
T008
First Name
Last Name
Back
Submit Teacher

13. Admin


Add New Admin
Admin Name
Email
Password
Confirm Password
Back
Submit Admin


Edit Admin
Admin Name
Admin Utama
Email
admin@superadmin.com
New Password (kosongkan jika tidak ingin mengganti password)
Confirm New Password
Back
Update Admin


Logout
CORE
Dashboard
INTERFACE
Students Data
Category + Subject
Timeslot
Teacher
Admin
Admin List
Search Admin...
Add Admin
Email
Name
Password
Actions
admin@superadmin.com
Admin Utama
Hidden
Edit
Delete
admin1@admin.com
Admin 1
admin2@admin.com
Admin 2
admin3@admin.com
Admin 3
admin4@admin.com
Admin 4

C.2. The Code

kodenya terdiri atas bagian model, controller, view(blade), middleware yang akan diberikan di file github.