

# Chapter 5

*by* Muhammad Darlen Sava

---

**Submission date:** 01-Jul-2023 06:03AM (UTC+0800)

**Submission ID:** 2124924709

**File name:** Ch\_5.docx (5.27M)

**Word count:** 1709

**Character count:** 8884

## 1 CHAPTER 5

### IMPLEMENTATION AND TESTING

#### 5.1 Introduction

In this chapter the implementation of the system along with the testing will be discussed. The chapter will provide the looks and explanation of the coding and interfaces of the system's main functions. The testing that are done to the system such as Black box testing, White box testing and User testing will also be discussed.

#### 1 5.2 Coding of System Main Functions

Below are some of the main functions in the system

##### 5 5.2.1 Login Function

```
const ADMIN = 1;
const TEACHER = 2;
const STUDENT = 3;

public function store(LoginRequest $request) {
    $request->authenticate();
    $request->session()->regenerate();

    switch (auth()->user()->userType) {
        case self::ADMIN:
            return redirect('/dashboard-admin');
        case self::TEACHER:
            return redirect('/dashboard-guru');
        case self::STUDENT:
            return redirect('/dashboard-siswa');
        default:
            return redirect('/');
    }
}
```

Figure 5.1: Login Function for Users

### 5.2.2 Logout Function

```
public function destroy(Request $request)
{
    $userType = auth()->user()->userType;
    Auth::guard('web')->logout();

    $request->session()->invalidate();
    $request->session()->regenerateToken();

    switch ($userType) {
        case self::ADMIN:
            return redirect('/');
        case self::TEACHER:
            return redirect('/');
        case self::STUDENT:
            return redirect('/');
        default:
            return redirect('/');
    }
}
```

Figure 5.2: Logout Function for Users

### 5.2.3 Add Teacher Account Function (Admin)

```
public function storeTeacher(Request $request)
{
    $data = $request->validate([
        'username' => 'required|string|max:255',
        'password' => 'required|string|confirmed|min:8',
        'userType' => ['required', Rule::in([1, 2, 3])],
    ]);

    $data['password'] = Hash::make($request->password);

    $user = User::create($data);

    if ($request->userType === 2) {
        $teacherData = [
            'user_id' => $user->id,
        ];

        $teacher = new Teacher($teacherData);
        $teacher->user_id = $user->id;
        $teacher->save();
    }

    return Inertia::render('Admin/TambahAkunGuru');
}
```

Figure 5.3: Add Teacher Account Function for Users

### 5.2.4 Add Admin Account (Admin)

```
public function storeAdmin(Request $request)
{
    $data = $request->validate([
        'name' => 'required|string|max:255',
        'username' => 'required|string|max:255',
        'password' => 'required|string|confirmed|min:8',
        'userType' => ['required', Rule::in([1, 2, 3])],
    ]);

    $data['password'] = Hash::make($request->password);

    $user = User::create($data);

    if ($request->userType === 1) {
        $admin = new Admin();
        $admin->name = $request->name;
        $user->admin()->save($admin);
    }

    return Inertia::render('Admin/TambahAkunAdmin');
}
```

Figure 5.4: Add Admin Account Function for Admin

### 5.2.5 Add Student Account (Admin)

```

    public function storeStudent(Request $request)
    {
        $data = $request->validate([
            'username' => 'required|string|max:255',
            'password' => 'required|string|confirmed|min:8',
            'userType' => ['required', Rule::in([1, 2, 3])],
            'selectedYear' => 'required',
            'selectedSemester' => 'required',
            'selectedPeminatan' => 'required',
            'selectedClassroom' => 'required',
        ]);

        $data['password'] = Hash::make($request->password);

        $user = User::create($data);

        if ($request->userType === 3) {
            $studentData = [
                'user_id' => $user->id,
                'semester' => $request->selectedSemester,
                'academicYear' => $request->selectedYear,
                'peminatan' => $request->selectedPeminatan,
                'classroom_id' => $request->selectedClassroom,
            ];

            $student = Student::create($studentData);
        }

        return Inertia::render('Admin/TambahAkunSiswa');
    }

```

Figure 5.5: Add Student Account Function for Users

### 5.2.6 Display User Based on Roles Function (Admin)

```

public function getUser($type = null)
{
    $query = User::query();

    if ($type === 'admin') {
        $query->where('userType', 1)->with('admin');
    } elseif ($type === 'teacher') {
        $query->where('userType', 2)->with('teacher');
    } elseif ($type === 'student') {
        $query->where('userType', 3)->with('student');
    }

    $users = $query->get();

    foreach ($users as $user) {
        if ($user->student) {
            $user->name = $user->student->name;
            $user->nis_nisn = $user->student->nis . '/' . $user->student->nisn;
            $user->peminatan = $user->student->peminatan;
        } elseif ($user->teacher) {
            $user->name = $user->teacher->name;
            $user->nip = $user->teacher->nip;
            $user->teacherRole = $user->teacher->teacherRole;
        } elseif ($user->admin) {
            $user->name = $user->admin->name;
        }
    }

    return Inertia::render('Admin/ManajemenAkun', [
        'users' => $users,
        'filter' => $type
    ]);
}

```

Figure 5.6: Display User Function for Admin

### 5.2.7 Teacher List Function (Student)

```

public function getUser($type = null)
{
    $teachers = Teacher::with('user')->get();

    return Inertia::render('Student/DaftarGuru', ['teachers' => $teachers]);
}

```

Figure 5.7: Teacher List Function for Student

### 5.2.8 Update Profile Function (Student)

```

public function updateProfile(Request $request)
{
    $request->validate([
        'password' => ['nullable', 'string', 'min:8', 'confirmed'],
        'name' => ['nullable', 'string', 'max:255'],
        'parentName' => ['nullable', 'string', 'max:255'],
        'parentPhoneNum' => ['nullable', 'string', 'max:255'],
        'address' => ['nullable', 'string', 'max:255'],
        'dob' => ['nullable', 'date'],
        'gender' => ['nullable', 'string', 'max:255'],
        'nis' => ['nullable', 'integer'],
        'nisn' => ['nullable', 'integer'],
    ]);

    $user = auth()->user();
    if ($request->filled('password')) {
        $user->password = Hash::make($request->password);
        $user->save();
    }

    $student = $user->student;
    if ($request->filled('name') || $request->filled('parentName') || $request->filled('parentPhoneNum') || $request->filled('address') || $request->filled('dob') || $request->filled('gender') || $request->filled('nis') || $request->filled('nisn')) {
        $student->update($request->all());
    }

    return Inertia::render('Student/StudentProfile');
}

```

Figure 5.8: Update Profile Function for Student

### 5.2.9 View Profile (Student)

```
public function getProfile(Request $request)
{
    $user = auth()->user();
    $student = $user->student;

    return Inertia::render('Student/StudentProfile', [
        'student' => $student,
        'user' => $user
    ]);
}
```

Figure 5.9: View Profile Function for Student

### 5.2.10 View Tuition Record (Student)

```
public function getTuitionRecords()
{
    $user = auth()->user();
    $student = $user->student;
    $id = $student->id;

    $tuitionRecordsFromDatabase = TuitionRecord::where('student_id', $id)
        ->get();

    $months_string = ['Januari', 'Februari', 'Maret', 'April', 'Mei', 'Juni', 'Juli', 'Agustus', 'September', 'Oktober', 'November', 'Desember'];
    $months = range(1, 12);

    $tuitionRecords = collect($months)->map(function ($month) use ($tuitionRecordsFromDatabase, $months_string) {
        $record = $tuitionRecordsFromDatabase->firstWhere('month', (int) $month);

        if ($record) {
            return [
                'month' => $months_string[$record->month - 1],
                'paymentDate' => $record->paymentDate,
                'paymentAmount' => $record->paymentAmount,
                'paymentProof' => $record->paymentProof,
                'tuitionStatus' => [
                    'status' => $record->tuitionStatus
                ],
            ];
        } else {
            return [
                'month' => $months_string[$month - 1],
                'paymentDate' => null,
                'paymentAmount' => null,
                'paymentProof' => null,
                'tuitionStatus' => [
                    'status' => null
                ],
            ];
        }
    });

    return Inertia::render('Student/StudentSPP', [
        'tuitionRecords' => $tuitionRecords,
        'studentId' => $id,
        'academicYear' => $student->academicYear,
    ]);
}
```

Figure 5.10: View Tuition Function for Student

### 5.2.11 Save Tuition Record (Student)

```
public function saveTuitionRecord(Request $request)
{
    $user = auth()->user();
    $student = $user->student;
    $id = $student->id;

    $month = $request->input('month');

    $tuitionRecord = TuitionRecord::where('student_id', $id)
        ->where('month', $month)
        ->first();

    if (!$tuitionRecord) {
        $tuitionRecord = new TuitionRecord;
        $tuitionRecord->student_id = $id;
        $tuitionRecord->month = $month;
    }

    if ($request->hasFile('paymentProof')) {
        $file = $request->file('paymentProof');
        $filename = time() . '.' . $file->getClientOriginalExtension();
        $file->move(public_path('uploads'), $filename);
        $tuitionRecord->paymentProof = $filename;
    }

    $tuitionRecord->paymentDate = $request->input('paymentDate');
    $tuitionRecord->paymentAmount = $request->input('paymentAmount');

    $unpaidStatus = TuitionStatus::where('status', 'Unpaid')->first();
    if ($unpaidStatus) {
        $tuitionRecord->tuitionStatus_id = $unpaidStatus->id;
    } else {
        $unpaidStatus = new TuitionStatus;
        $unpaidStatus->status = 'Unpaid';
        $unpaidStatus->save();
        $tuitionRecord->tuitionStatus_id = $unpaidStatus->id;
    }

    $tuitionRecord->save();

    return response()->json(['message' => 'Tuition record saved successfully']);
}
```

Figure 5.11: Save Tuition Function for Student

### 5.2.12 View Academic Report (Student)



```

public function viewAcademicReport($id)
{
    $user = auth()->user();
    $student = $user->student;
    $id = $student->id;

    $academicReport = $id->academicReport;

    $subjectMarks = $academicReport->subjectMarks;

    $skillMarks = [];
    $knowledgeMarks = [];
    foreach ($subjectMarks as $subjectMark) {
        $skillMark = $subjectMark->skillMark;
        $knowledgeMark = $subjectMark->knowledgeMark;
        array_push($skillMarks, $skillMark);
        array_push($knowledgeMarks, $knowledgeMark);
    }

    return Inertia::render('Student/StudentRapon', [
        'academicReport' => $academicReport,
        'subjectMarks' => $subjectMarks,
        'skillMarks' => $skillMarks,
        'knowledgeMarks' => $knowledgeMarks,
    ]);
}

```

Figure 5.7: View Academic Report Function for Student

### 5.2.13 View Student (Teacher)

```

public function getUserPenilaian($type = null)
{
    $students = Student::with('user')->get()->map(function ($student) {
        $student->nis_nisn = $student->nis . '/' . $student->nisn;
        $student->classroom_id = $student->classroom_id;
        return $student;
    });

    $classrooms = Classroom::all();
    return Inertia::render('Teacher/Penilaian', ['students' => $students, 'classrooms' => $classrooms]);
}

```

```

    public function getUser($type = null)
    {
        $students = Student::with('user')->get()->map(function ($student) {
            $student->nis_nisn = $student->nis . '/' . $student->nisn;
            $student->classroom_id = $student->classroom_id;
            return $student;
        });

        $classrooms = Classroom::all();
        return Inertia::render('Teacher/DaftarSiswa', ['students' => $students, 'classrooms' => $classrooms]);
    }

    public function getUserSPP($type = null)
    {
        $students = Student::with('user')->get()->map(function ($student) {
            $student->nis_nisn = $student->nis . '/' . $student->nisn;
            $student->classroom_id = $student->classroom_id;
            return $student;
        });

        $classrooms = Classroom::all();
        return Inertia::render('Teacher/SPP', ['students' => $students, 'classrooms' => $classrooms]);
    }

    public function getUserAbsensi($type = null)
    {
        $students = Student::with('user')->get()->map(function ($student) {
            $student->nis_nisn = $student->nis . '/' . $student->nisn;
            $student->classroom_id = $student->classroom_id;
            return $student;
        });

        $classrooms = Classroom::all();
        return Inertia::render('Teacher/Absensi', ['students' => $students, 'classrooms' => $classrooms]);
    }

    public function getUserRapor($type = null)
    {
        $students = Student::with('user')->get()->map(function ($student) {
            $student->nis_nisn = $student->nis . '/' . $student->nisn;
            $student->classroom_id = $student->classroom_id;
            return $student;
        });

        $classrooms = Classroom::all();
        return Inertia::render('Teacher/TeacherRapor', ['students' => $students, 'classrooms' => $classrooms]);
    }
}

```

Figure 5.13: View Student Function for Teacher in various pages

#### 5.2.14 View Input Mark (Teacher)

```

public function getInputMarksPage($studentId)
{
    $student = Student::with('user')->find($studentId);
    $subjects = Subject::all(); // Fetch all subjects

    return Inertia::render('Teacher/InputPenilaian', ['student' => $student, 'subjects' => $subjects]);
}

```

Figure 5.14: View Input Mark Function for Teacher

#### 5.2.15 View Tuition Record (Teacher)

```

public function getTuitionRecords($id)
{
    $student = Student::find($id);

    $tuitionRecordsFromDatabase = TuitionRecord::with('tuitionStatus')->where('student_id', $id)->get();

    $months_string = ['Januari', 'Februari', 'Maret', 'April', 'Mei', 'Juni', 'Juli', 'Agustus', 'September', 'Oktober', 'November', 'Desember'];
    $months = range(1, 12);
    $tuitionRecords = collect($months)->map(function ($month) use ($tuitionRecordsFromDatabase, $months_string) {
        $record = $tuitionRecordsFromDatabase->firstWhere('month', $month);

        if ($record) {
            return [
                'month' => $months_string[$record->month - 1],
                'paymentDate' => $record->paymentDate,
                'paymentAmount' => $record->paymentAmount,
                'paymentProof' => $record->paymentProof,
                'tuitionStatus' => [
                    'status' => $record->tuitionStatus->status ?? null,
                ],
                'tuitionStatus_id' => $record->tuitionStatus->id ?? null,
            ];
        } else {
            return [
                'month' => $months_string[$month - 1],
                'paymentDate' => null,
                'paymentAmount' => null,
                'paymentProof' => null,
                'tuitionStatus' => [
                    'status' => null
                ],
                'tuitionStatus_id' => null,
            ];
        }
    });

    // get tuition status
    $tuitionStatuses = TuitionStatus::all();

    return Inertia::render('Teacher/ViewSPP', [
        'tuitionRecords' => $tuitionRecords,
        'studentId' => $id,
        'studentName' => $student->name,
        'tuitionStatuses' => $tuitionStatuses,
    ]);
}

```

Figure 5.15: View Tuition Record Function for Teacher

### 5.2.16 Update Tuition Status (Teacher)

```

public function updateTuitionStatus(Request $request, $id, $month)
{
    $request->validate([
        'tuitionStatus_id' => 'required|exists:tuition_statuses,id',
    ]);

    $tuitionRecord = TuitionRecord::where('student_id', $id) ->where('month', $month)->first();

    if (!$tuitionRecord) {
        return response()->json(['message' => 'Tuition record not found'], 404);
    }

    $tuitionRecord->tuitionStatus_id = $request->input('tuitionStatus_id');
    $tuitionRecord->save();

    return redirect()->back();
}

```

Figure 5.16: View Tuition Status Function for Teacher

### 5.2.17 View Profile Page (Teacher)

```
public function getProfile(Request $request)
{
    $user = auth()->user();
    $teacher = $user->teacher;

    return Inertia::render('Teacher/TeacherProfile', [
        'teacher' => $teacher,
        'user' => $user
    ]);
}
```

Figure 5.17: View Profile Function for Teacher

### 5.2.18 View Edit Profile Page (Teacher)

```
public function getProfileEdit(Request $request)
{
    $user = auth()->user();
    $teacher = $user->teacher;

    return Inertia::render('Teacher/TeacherProfileEdit', [
        'teacher' => $teacher,
        'user' => $user
    ]);
}
```

Figure 5.18: View Edit Profile Function for Teacher

### 5.2.19 Update Profile (Teacher)

```
public function updateProfile(Request $request)
{
    You, 2 minutes ago • Uncommitted changes
    $request->validate([
        'password' => ['nullable', 'string', 'min:8', 'confirmed'],
        'name' => ['nullable', 'string', 'max:255'],
        'phoneNum' => ['nullable', 'string', 'max:255'],
        'address' => ['nullable', 'string', 'max:255'],
        'dob' => ['nullable', 'date'],
        'gender' => ['nullable', 'string', 'max:255'],
        'teacherRole' => ['nullable', 'string', 'max:255'],
        'nip' => ['nullable', 'integer'],
    ]);

    $user = auth()->user();
    if ($request->filled('password')) {
        $user->password = Hash::make($request->password);
        $user->save();
    }

    $teacher = $user->teacher;
    if ($request->filled('name') || $request->filled('phoneNum') || $request->filled('address') || $request->filled('dob') ||
        $request->filled('gender') || $request->filled('teacherRole') || $request->filled('nip')) {
        $teacher->update($request->all());
    }

    return Inertia::render('Teacher/TeacherProfile');
}
```

Figure 5.19: Update Profile Function for Teacher

### 5.2.20 Save Mark (Teacher)

```
public function saveMark(Request $request)
{
    $validatedData = $request->validate([
        'kkm' => 'required|integer',
        'nilai' => 'required|integer',
        'deskripsi' => 'required|string',
        'aspek' => 'required|string',
        'subject_id' => 'required|integer',
        'student_id' => 'required|integer',
    ]);

    $subjectMark = new SubjectMark;
    $subjectMark->kkm = $validatedData['kkm'];
    $subjectMark->subject_id = $validatedData['subject_id'];
    $subjectMark->teacher_id = auth()->user()->id;
    $subjectMark->save();

    if ($validatedData['aspek'] == 'skill') {
        $mark = new SkillMark;
        $mark->skillMark = $validatedData['nilai'];
        $mark->skillMarkPredicate = $this->getPredicate($validatedData['nilai']);
        $mark->skillMarkDescription = $validatedData['deskripsi'];
    } else {
        $mark = new KnowledgeMark;
        $mark->knowledgeMark = $validatedData['nilai'];
        $mark->knowledgePredicate = $this->getPredicate($validatedData['nilai']);
        $mark->knowledgeDescription = $validatedData['deskripsi'];
    }

    $mark->subjectMark_id = $subjectMark->id;
    $mark->save();

    return response()->json(['message' => 'Mark saved successfully']);
}
```

Figure 5.20: Save Mark Function for Teacher

### 5.2.21 View Timetable (Teacher)

```
public function getTimetable(Request $request, $classId)
{
    $classrooms = Classroom::all();

    $timetable = Timetable::where('class_id', $classId)
        ->with([
            'hour1Subject', 'hour2Subject', 'hour3Subject', 'hour4Subject',
            'hour5Subject', 'hour6Subject', 'hour7Subject', 'hour8Subject'
        ]->get());

    $subjects = Subject::all();

    $subjects->push((object)[
        'id' => -1,
        'subjectName' => 'Break Time'
    ]);

    return Inertia::render('Teacher/ZadwalPelajaran', ['classrooms' => $classrooms, 'timetable' => $timetable, 'subjects' => $subjects]);
}
```

Figure 5.21: View Timetable Function for Teacher

### 5.2.22 View Timetable (Student)

```
public function getTimetableStudent(Request $request, $classId)
{
    $classrooms = Classroom::all();

    $timetable = Timetable::where('class_id', $classId)
        ~>with([
            'hour1Subject', 'hour2Subject', 'hour3Subject', 'hour4Subject',
            'hour5Subject', 'hour6Subject', 'hour7Subject', 'hour8Subject'
        ]->get();

    $subjects = Subject::all();

    $subjects->push((object)[
        'id' => -1,
        'subjectName' => 'Break Time'
    ]);

    return Inertia::render('Student/JadwalPelajaran', ['classrooms' => $classrooms, 'timetable' => $timetable, 'subjects' => $subjects]);
}
```

Figure 5.22: View Timetable Function for Student

### 5.2.23 Search Function (General)

The search function is implemented in the front end. Below is one of the example of the implementation.

```
const filters = ref({
  classroom_id: { value: null, matchMode: FilterMatchMode.EQ },
  semester: { value: null, matchMode: FilterMatchMode.CONTAINS },
  academicYear: { value: null, matchMode: FilterMatchMode.CONTAINS },
  global: { value: null, matchMode: FilterMatchMode.CONTAINS },
  name: { value: null, matchMode: FilterMatchMode.STARTS_WITH },
  status: { value: null, matchMode: FilterMatchMode.EQUALS },
  verified: { value: null, matchMode: FilterMatchMode.EQUALS },
});
```

Figure 5.23: Search Function in General

## 5.3 Interfaces of System Main Functions

Below are some of the main functions interfaces of the system.

### 5.3.1 Login Page

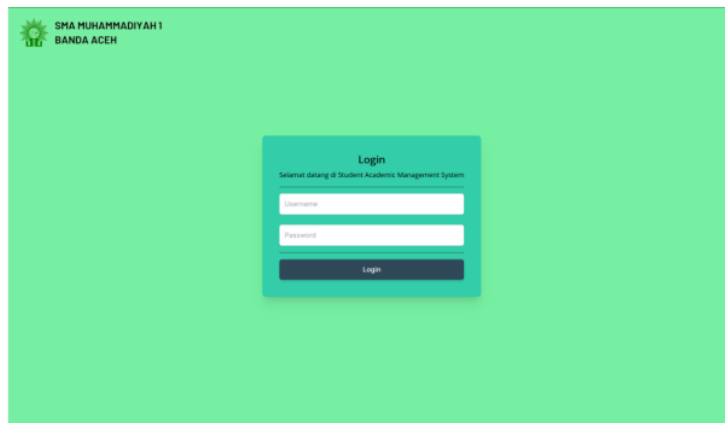


Figure 5.24: Login Page for All Users

### 5.3.2 Admin Dashboard Page

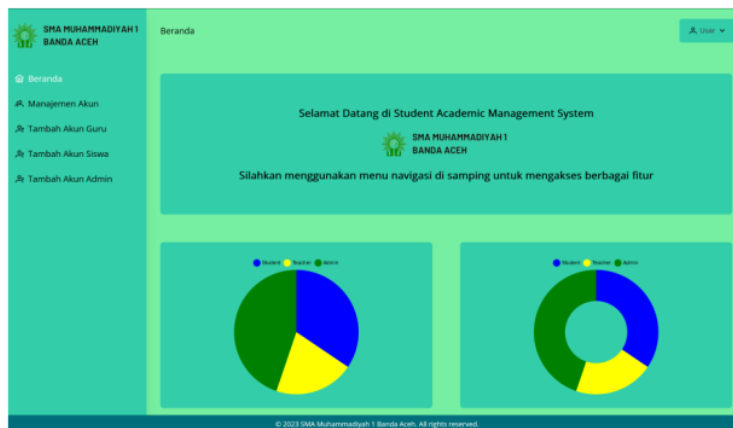


Figure 5.25: Dashboard Page for Admin

### 5.3.3 Account Management Page

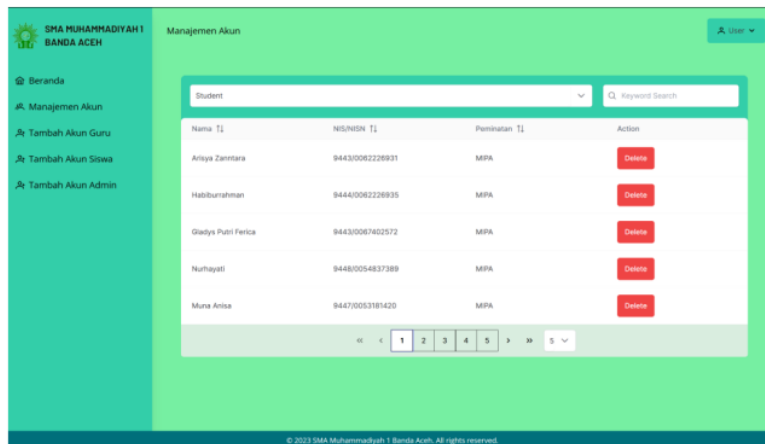


Figure 5.26: Account Management Page for Admin

#### 5.3.4 Add Teacher Account Page

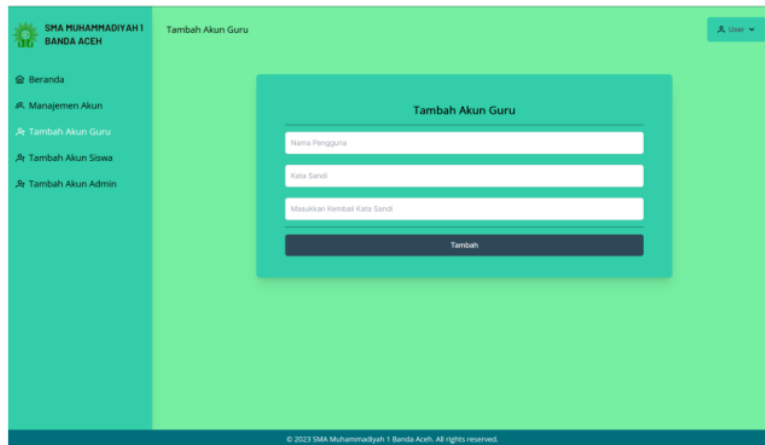


Figure 5.27: Add Teacher Page for Admin

#### 5.3.5 Add Student Account Page



Figure 5.28: Add Student Page for Admin

### 5.3.6 Add Admin Account Page

Figure 5.29: Add Admin Page for Admin

### 5.3.7 Student Dashboard Page

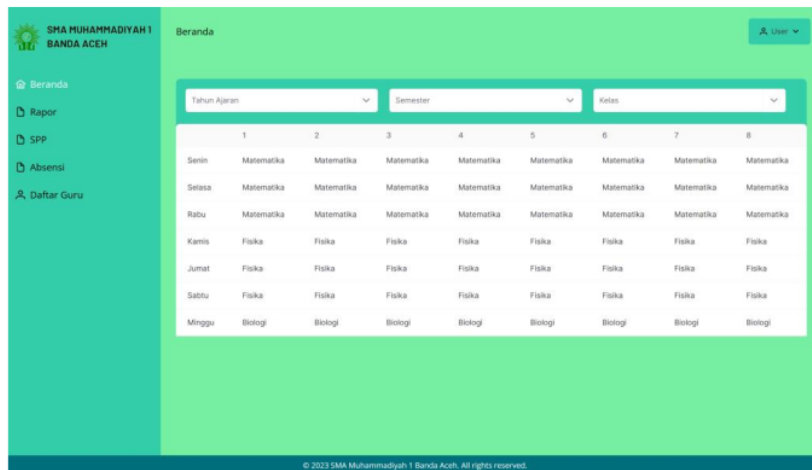


Figure 5.30: Dashboard Page for Student

### 5.3.8 Student Academic Report Page

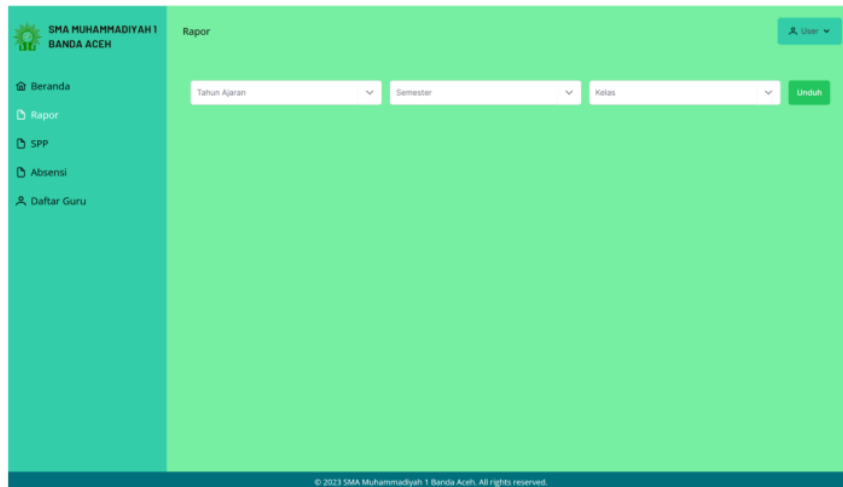
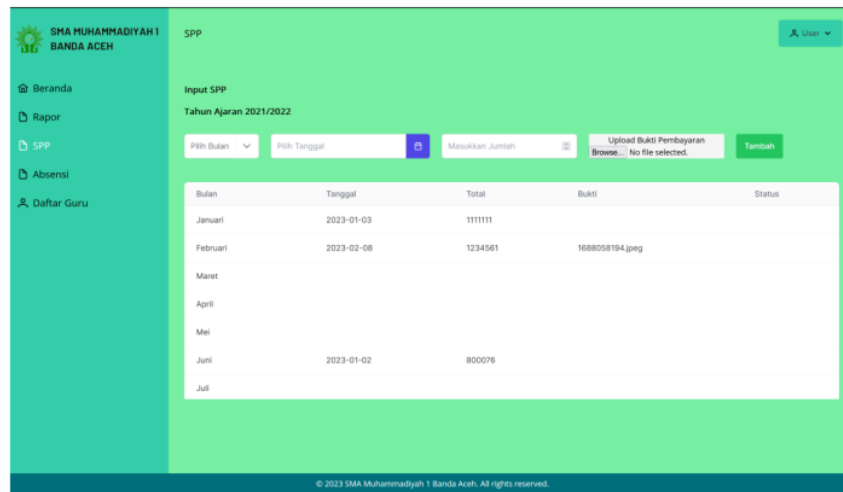


Figure 5.31: Academic Report Page for Student





### 5.3.9 Student Tuition Page



SMA MUHAMMADIYAH 1  
BANDA ACEH

SPP

Input SPP  
Tahun Ajaran 2021/2022

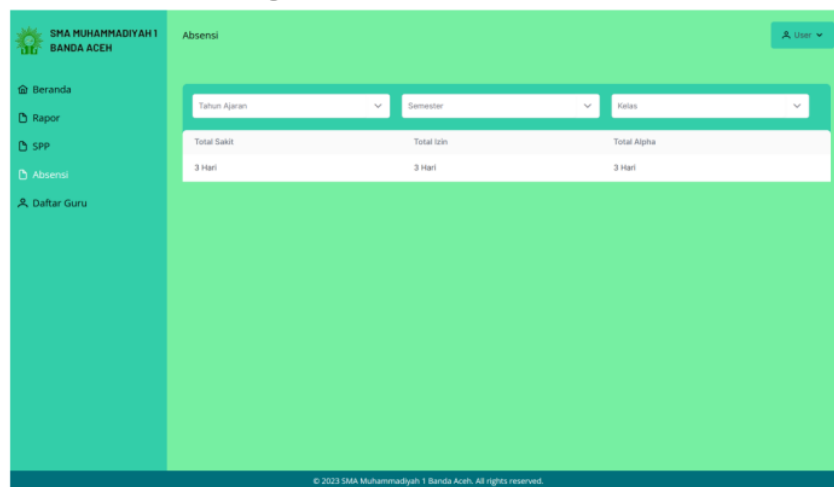
Pilih Bulan ▼ Pilih Tanggal  Masukkan Jumlah  Upload Bukti Pembayaran  No file selected. 

Bulan	Tanggal	Total	Bukti	Status
Januari	2023-01-03	1111111		
Februari	2023-02-08	1234561	1688058194.png	
Maret				
April				
Mei				
Juni	2023-01-02	800076		
Juli				

© 2023 SMA Muhammadiyah 1 Banda Aceh. All rights reserved.

Figure 5.32: Tuition Page for Student

### 5.3.10 Student Attendance Page



SMA MUHAMMADIYAH 1  
BANDA ACEH

Absensi

Tahun Ajaran ▼ Semester ▼ Kelas ▼

Total Sakit	Total Izin	Total Alpha
3 Hari	3 Hari	3 Hari

© 2023 SMA Muhammadiyah 1 Banda Aceh. All rights reserved.

Figure 5.33: Attendance Page for Student

### 5.3.11 Teacher List Page

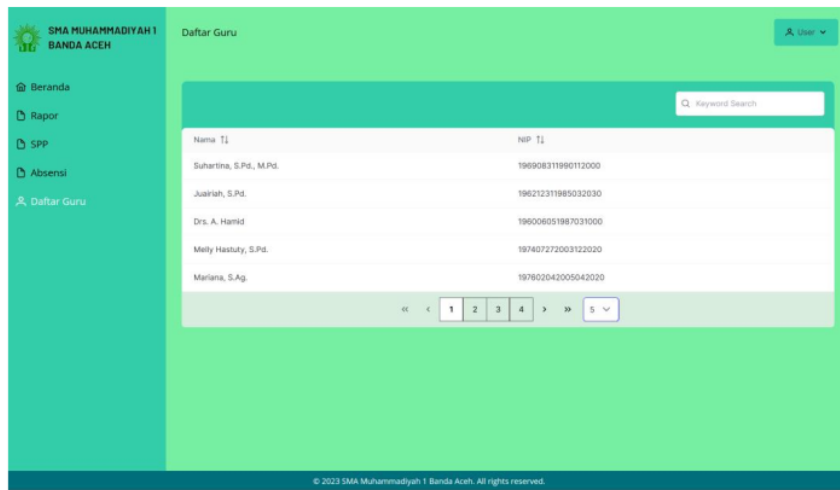


Figure 5.34: Teacher List Page for Student

### 5.3.12 Teacher Manage Academic Report Page

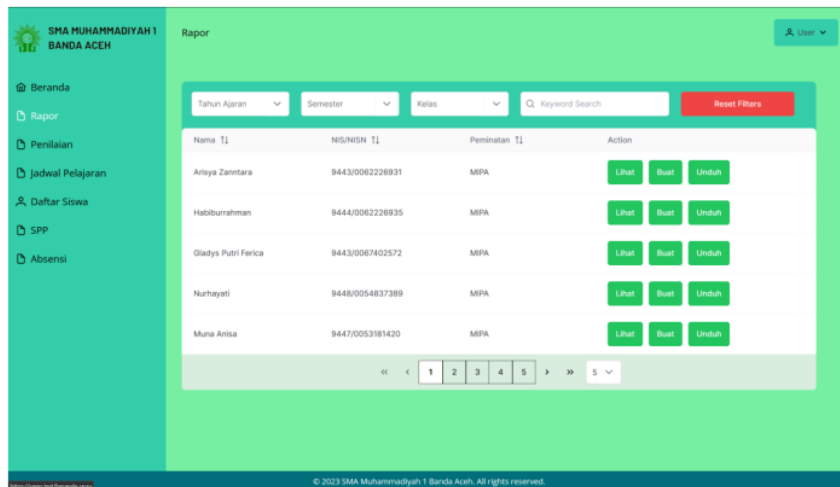


Figure 5.35: Manage Academic Record Page for Teacher

### 5.3.13 Teacher Manage Marks Page

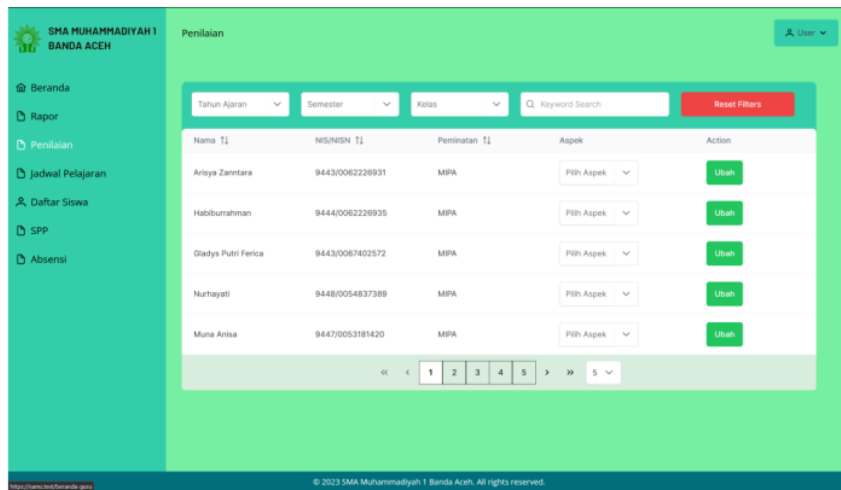


Figure 5.36: Manage Marks Page for Teacher

### 5.3.14 Teacher Manage Timetable Page

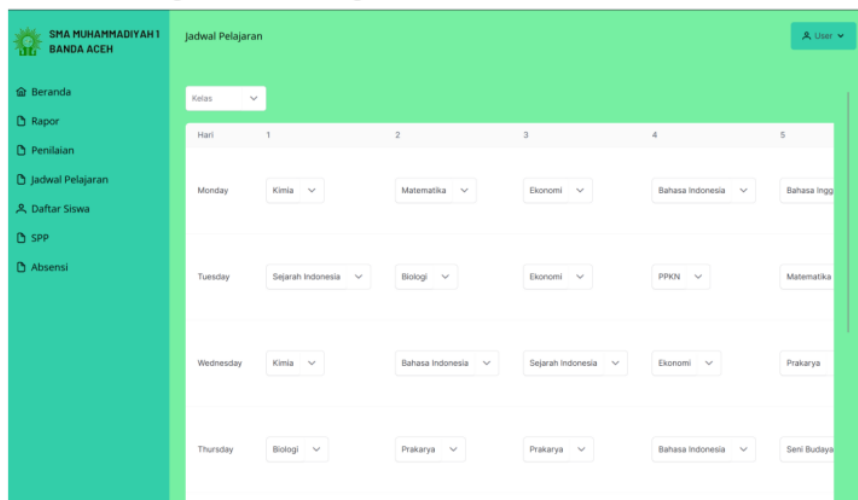


Figure 5.37: Manage Timetable Page for Teacher

### 5.3.15 Teacher Student List Page

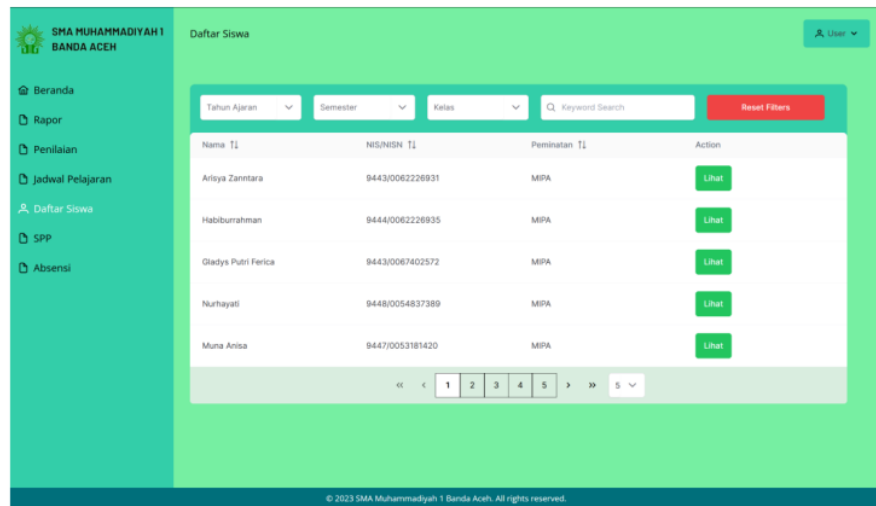


Figure 5.38: Student List Page for Teacher

### 5.3.16 Teacher Manage Tuition Page

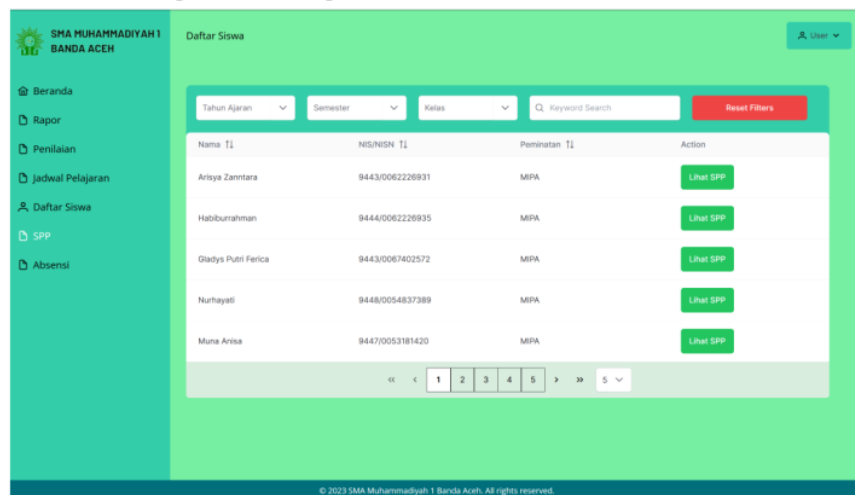


Figure 5.39: Manage Tuition Page for Teacher

### 5.3.17 Teacher Manage Attendance Page

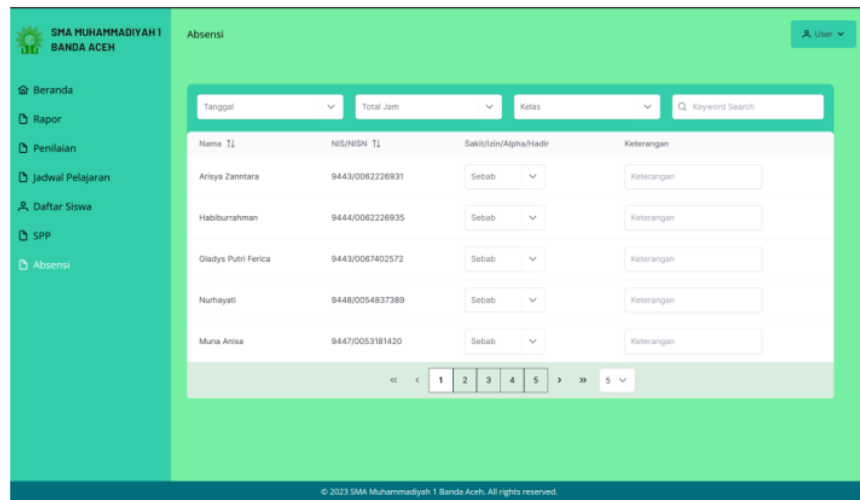


Figure 5.40: Manage Attendance Page for Teacher

## 5.4 Testing

There is a couple testing method that will be done to the system. They are black box testing, white box testing and user testing. Testing is an important process of the system development since it can identify errors and bugs. This establish the system to be as close as possible to the requirement.

### 5.4.1 Black box Testing

Black box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. In black box testing, the test done is more focused towards the input and output of the system without knowing how the application receive the input and produce the output. Black box testing also done from the perspective of the user to ensure that the functions works and follow the stated requirements

#### **5.4.1.1 System Flow**

The system flow of an application refers to the sequence of events or operations that a system undergoes when performing specific tasks. In black box testing, testers observe and evaluate whether the system behaves as expected when performing a particular sequence of actions.

#### **5.4.1.2 Input Output Verification**

Input Output Verification in black box testing focuses on testing the system's response to given inputs and whether the output is as expected. Testers provide various types of valid and invalid inputs and verify if the system produces the correct and expected output. This process helps in identifying any discrepancies in the system's functionality.

#### **5.4.1.3 Error Message**

Error message testing in black box testing focuses on the system's response to erroneous conditions or inputs. This involves deliberately inputting incorrect or invalid data and verifying the system's error messages. The objective is to ensure that the system correctly detects errors and handles them

### **5.4.2 <sup>8</sup> White box Testing**

In white box testing, the tester that conducts the testing has information on the system's internal <sup>6</sup> structure, design, and implementation. The goal of white



box testing is to verify a system's internal operations, focusing on internal testing of code structures, code flows, and internal system logic. Some of example of this type of testing are provided below in the User Testing subchapter.

### 5.4.3 User Testing

User Testing will be performed by real users to evaluate the system. User Testing provides direct input on how real users use a system, thereby helping to understand where the system stands from the user's perspective. The user testing results show average user satisfaction with the system. The user testing has been done to the system, and the result of the test can be seen in the table 5.1.

Table 5.1 User Testing Result

User Acceptance Testing Result	
Questions	Average Satisfaction
1. Does the system effectively perform all the functions requirements?	8.5/10
2. Is the user interface of the system intuitive and easy to navigate?	9/10
3. Does the system handle errors effectively and provide meaningful error messages?	7.5/10
4. Is the system's performance satisfactory under various loads?	8/10
5. Are the system's security measures effective and reliable?	8/10

Table 5.2 Admin Add Student Account

Tester Name	RAHMAWATI
Date	23 June 2023
Module	Account Module
<b>Instruction</b>	
1. Login with Admin Account	
2. Click "Tambah Akun Siswa"	
3. Enter the required input	
4. Click "Tambah"	

5. Logout by clicking the top right menu and click “Keluar” button
<b>Expected Result</b> <ol style="list-style-type: none"> <li>1. Should be able to logged in as Admin</li> <li>2. Should be able to view the “Tambah Akun Siswa” Page</li> <li>3. Should be able to input all the required fields</li> <li>4. Should be able to create the student account with the provided input</li> <li>5. Should be able to logged out from the sytem</li> </ol>
<b>Actual Result</b> <ol style="list-style-type: none"> <li>1. Successfully logged in as Admin</li> <li>2. Successfully view the “Tambah Akun Siswa” Page</li> <li>3. Successfully input all the required fields</li> <li>4. Successfully create the student account with the provided input</li> <li>5. Successfully logged out from the system</li> </ol>

Table 5.3 Admin Add Teacher Account

<b>Tester Name</b>	RAHMAWATI
<b>Date</b>	23 June 2023
<b>Module</b>	Account Module
<b>Instruction</b> <ol style="list-style-type: none"> <li>1. Login with Admin Account</li> <li>2. Click “Tambah Akun Guru”</li> <li>3. Enter the required input</li> <li>4. Click “Tambah”</li> <li>5. Logout by clicking the top right menu and click “Keluar” button</li> </ol>	
<b>Expected Result</b> <ol style="list-style-type: none"> <li>1. Should be able to logged in as Admin</li> <li>2. Should be able to view the “Tambah Akun Guru” Page</li> <li>3. Should be able to input all the required fields</li> <li>4. Should be able to create the student account with the provided input</li> <li>5. Should be able to logged out from the system</li> </ol>	
<b>Actual Result</b> <ol style="list-style-type: none"> <li>1. Successfully logged in as Admin</li> <li>2. Successfully view the “Tambah Akun Guru” Page</li> <li>3. Successfully input all the required fields</li> <li>4. Successfully create the student account with the provided input</li> <li>5. Successfully logged out from the system</li> </ol>	

Table 5.4 Teacher Marking Student

<b>Tester Name</b>	YULIANA
<b>Date</b>	23 June 2023
<b>Module</b>	Report Module
<b>Instruction</b> <ol style="list-style-type: none"> <li>1. Login with Teacher Account</li> <li>2. Click “Penilaian”</li> <li>3. Select the filter to view the desired student</li> <li>4. Click the “Nilai” button</li> <li>5. Input the required field</li> <li>6. Click the “Save” button</li> <li>7. Logout by clicking the top right menu and click “Keluar” button</li> </ol>	
<b>Expected Result</b>	

<ol style="list-style-type: none"> <li>1. Should be able to logged in as Teacher</li> <li>2. Should be able to view the "Penilaian" Page</li> <li>3. Should be able to view the desired student</li> <li>4. Should be able to view the input mark page</li> <li>5. Should be able to save the mark for the student</li> <li>6. Should be able to logged out from the system</li> </ol>
<p style="text-align: center;"><b>Actual Result</b></p> <ol style="list-style-type: none"> <li>1. Successfully logged in as Teacher</li> <li>2. Successfully view the "Penilaian" Page</li> <li>3. Successfully view the desired student</li> <li>4. Successfully view the input mark page</li> <li>5. Successfully save the mark for the student</li> <li>6. Successfully logged out from the system</li> </ol>

Table 5.5 Student View Academic Report

<b>Tester Name</b>	ARISYA ZANNTARA
<b>Date</b>	23 June 2023
<b>Module</b>	Report Module
<p style="text-align: center;"><b>Instruction</b></p> <ol style="list-style-type: none"> <li>1. Login with Student Account</li> <li>2. Click "Rapor"</li> <li>3. Select the filter to preview the desired Academic Report</li> <li>4. Click "Unduh"</li> <li>5. Logout by clicking the top right menu and click "Keluar" button</li> </ol>	
<p style="text-align: center;"><b>Expected Result</b></p> <ol style="list-style-type: none"> <li>1. Should be able to logged in as Student</li> <li>2. Should be able to view the "Rapor" Page</li> <li>3. Should be able to view the desired Academic Report</li> <li>4. Should be able to download the Academic Report</li> <li>5. Should be able to logged out from the system</li> </ol>	
<p style="text-align: center;"><b>Actual Result</b></p> <ol style="list-style-type: none"> <li>1. Successfully logged in as Student</li> <li>2. Successfully view the "Rapor" Page</li> <li>3. Successfully view the desired Academic Report</li> <li>4. Successfully download the Academic Report</li> <li>5. Successfully logged out from the system</li> </ol>	

## 5.5 Chapter Summary

This chapter conclude the implementation and testing of the Student Academic Management System. All of the codes and interfaces main function are discussed and showed thoroughly. The testing done which are Black box testing, White box testing and User testing was also provided and explained.

# Chapter 5

## ORIGINALITY REPORT

11%  
SIMILARITY INDEX

5%  
INTERNET SOURCES

0%  
PUBLICATIONS

10%  
STUDENT PAPERS

## PRIMARY SOURCES

1	Submitted to Universiti Teknologi Malaysia Student Paper	3%
2	Submitted to KDU College Sdn Bhd Student Paper	2%
3	Submitted to University of Stirling Student Paper	2%
4	Submitted to Mazoon University College Student Paper	1%
5	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	1%
6	Submitted to Midlands State University Student Paper	1%
7	diginole.lib.fsu.edu Internet Source	1%
8	nou.edu.ng Internet Source	1%
9	in.jooble.org Internet Source	1%

---

Exclude quotes      Off

Exclude matches      < 1%

Exclude bibliography      On