APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 5 No. 2 (YEAR) 598-615
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

Sports, Events and Facilities Rental System for Cameron Highland District Council

Goh Jia Xing[1], Shahreen Kasim[2]*

[1,2] *Fakulti Sains Komputer dan Teknologi Maklumat,*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: shahreen@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2024.05.02.032

Article Info

Received: 13 June 2024
Accepted: 28 September 2024
Available online: 15 December 2024

Keywords

Rental system, sport rental, event rental, facility rental

Abstract

The sports, events and facilities rental system is a rental system that is built for Cameron Highland District Council. The aim is to improve the rental process because they still use traditional ways to record the rental details manually. The system is developed to make the rental process easier. Users can rent the sports, events and facilities anywhere, anytime. Furthermore, the administrator can update and check the rental record easily. There are some problems with using traditional ways to record rental data which are data security, accuracy and reliability, and accessibility. The initiative of this system is motivated by the realization that the current system is frequently disjointed and lacks the cohesiveness required to offer users a seamless and practical experience. This web-based platform, designed with user-friendliness in mind, enables rental in a short time anywhere, anytime.

1. Introduction

A rental system is that a company or organization provides short-term leasing of venues and facilities for a specific time with a fee to their customers using the Internet [1]. Effective management of sports, events and facilities has become more and more important for local governing bodies in the modern era. Streamlining operations and increasing community engagement have been made possible by integrating technology into administrative processes [2]. Tourism is positively impacted when large sporting events are hosted, and excellent management is upheld within the organization [3]. The project aims to offer a centralized system for booking, scheduling, and tracking the use of sporting facilities through the deployment of a digital platform. In addition, it will streamline the process of organizing and coordinating events, making the district's calendar of events more accessible and well-organized.

The current rental system for sports, events and facilities administered by the Cameron Highland District Council encountered operational inefficiencies, affecting both users and administrators. The lack of a well-organized system causes issues and hinders the overall efficiency of the renting process. The development of a sports, events and facilities rental system is crucial to addressing these issues. Firstly, the design approach is critical to creating a robust rental system. The absence of a well-defined structure jeopardizes the adaptability and maintainability of the system. Furthermore, the traditional system is characterized by inefficiency because it needs to record the rentals one by one. This may lead to time-consuming.
Moreover, the traditional system is exposed to security vulnerabilities, risking the confidentially sensitive data of users to be leaked. The development of sports, events and facilities rental system is crucial to addressing these issues.

The objective of this proposed system is to develop a well-rounded and user-friendly rental system catering for the target users, which are residents in Cameron Highland. This system aims to offer seamless

functionality for sports, events and facilities rental. Users can conveniently perform these functions using their computers connected to the Internet.

Furthermore, the proposed system is designed to be utilized by administrators responsible for managing the overall rental process. The administrator dashboard will be provided to them, allowing comprehensive control over the entire system. Administrators can configure the system to add and drop the sports, events and facilities. Moreover, the administrator also can view the user rental record.

2. Related Work

This section provided a literature review on topics related to sports, events and facilities rental system. The literature review aims to gain a comprehensive understanding and discuss the project background of the proposed system.

2.1 Rental System

A rental is an agreement to rent something for a specific period. A system refers to a collection of computer equipment and programs that are used together for a specific purpose. The rental system acts as a digital intermediary between individuals who require something (such as a space to rent) and those who have it (such as the Cameron Highland District Council). It is not just about renting, but rather a comprehensive set of tools and technology that is designed to make the entire process smooth, efficient, and accessible for everyone involved.

2.2 Personal Home Page (PHP)

Personal Home Page, also referred to as Hypertext Preprocessor (PHP), was developed by Rasmus Lerdorf in 1994 and made available to the public for use in 1995 [4]. Many developers use PHP, an open-source server-side programming language, to create websites. Additionally, it is a general-purpose language that may be used to create a wide range of applications, such as GUIs (graphical user interfaces). Because PHP is a server-side programming language, it creates, updates, deletes, and manages dynamic information via database connections, XML documents, pictures, PDF files, and other forms. As such, most of the work is done on the server side. Programs can also be run straight from the command line using PHP. System management tasks are typically performed by command-line scripts.

There are some advantages of using PHP. The first is cross-platform. PHP can run on every platform whether it is MacOS, Windows or Linux. Furthermore, PHP syncs with all databases. We can easily connect PHP to all databases, relational and non-relational in no time. Moreover, websites created using PHP load quickly. In the modern era of fast internet speeds, people like websites that load quickly; they would never tolerate a website that takes a long time to load, which is why consumers who are looking for speed always favour websites built with PHP.

2.3 Laravel

Taylor Otwell created Laravel in July 2011[5]. Laravel is one of the greatest and most advanced PHP frameworks available. Its features include enhanced software quality, streamlined authentication, effortless routing, convenient access, and increased website framework power [6]. The framework application Laravel offers a beautiful syntax and many features, including security, password storage, password reminders and resets, encryption, and validation. The popular MVC design pattern is applied to interactive software system designs. The MVC methodology operates by dividing the primary elements—data manipulation (Model), display/interface (View), and process (Controller)—into smaller, more easily built parts [7]. The Model interacts with the database. View is the interface for Users. It includes all the content that the user can see on the screen. The controller holds all of the business logic and aids in tying the Model and View together it is also referred to as the "MVC application's heart."

The browser sends a request command to the Controller first. The Controller communicates with the Model to send and receive data. To render the data, the Controller speaks with the View. The View's sole responsibility is to observe the delivery of information, not its final presentation. It will be a dynamic HTML file that renders information in response to input from the Controller. The Controller will then display the information to the user when the View processes its final presentation to it. The View and Model never interact, they need the help of the Controller to communicate with each other.

Strong database features offered by Laravel include the ORM (Object Relational Mapper) Eloquent as well as integrated seeders and migrations for databases. Application development can be accelerated by bootstrapping new models, controllers, and other components using the command-line tool Artisan.

2.4 Comparison with the Existing System

This section will make a comparison of the functionality of the selected existing rental system with the proposed system. Table 1 below presents the results of the comparison among the three selected systems with the proposed system.

Table 1 *Comparison between 3 existing systems and the proposed system*

Features	Pitchbooking[8]	Spacebase[9]	SportsBooker[10]	Proposed system
Calendar view for availability	√	√	√	√
Search by location, type, date, time	√	√	√	√
View booked slots and facility usage	√	√	√	√
Send booking confirmations & reminders	√	√	√	√
Track equipment and facility availability	√	√	√	√
Manage rental items	√	√	×	√
Multiple payment options	√	√	√	√
Login authentication	√	√	√	√
Multi-level user permission	√	√	×	√

3. Methodology

The methodology that is chosen to help in developing the system is the agile scrum model methodology. The Agile Scrum methodology combines the Scrum framework with the Agile philosophy. Agile stands for "incremental" which enables groups to work on projects in manageable chunks [11]. It is beneficial for a project team to complete their tasks systematically as well as improve the quality of the final output. During the initial phase of this project, an interview among the stakeholders and developer was conducted before the project started to develop. Including the executive general manager, Mr. Nazlan Mohamad, and Mrs. Norfai'eza Zainuren from MZR Global Sdn Bhd. These interviews will be conducted in every sprint session that is mentioned in the agile scrum methodology model. There are five phases in the project development process, such as initiation phase, planning and estimates phase, implementation phase, review and retrospective phase and release phase. The agile scrum model is shown in Figure 1 below.

Fig. 1 Agile Scrum Model [12]

Applied Information Technology and Computer Science Vol. 5 No. 2 (2024) p. 598-615

3.1 System Development Workflow

In this section, there is a total of five phases including the agile scrum methodology model. Table 2 shows the tasks and outputs during the respective phases of project development.

Table 2 *System development workflow*

Phase	Task	Output
Initiation	- Define project vision and objective - Outline main aims, objectives, and results. - Form project backlog	- Project vision, objective, and backlog
Planning and estimates	- Structure work into Scrum sprints (2-4 weeks) - Create a product backlog (user stories ordered by priority)	- Planned sprints, and prioritized product backlog.
Implementations	- Execute actions for desired project outcomes - Focus on completing tasks for the project goals and deliverables	- Desired project outcomes and completed tasks.
Review and Retrospective	- Evaluate progress and identify areas for development. - Conduct sprint review, retrospective and grooming. - Reflect, discuss ideas and plan improvement.	- Lessons learned, improvement, refined backlog. - Sprint Review meeting that occurs every two weeks.
Release	- Prepare and deliver final project outcomes to stakeholders. - Ensure successful project completion.	- User feedback through Google Forms.

3.2 Hardware Requirements

Table 3 below shows the hardware requirements for the project development process.

Table 3 *Hardware Requirements*

Hardware	Description
Laptop	The primary hardware needed to build the proposed system
Solid State Drive (SSD)	Hardware for storing every piece of data and information while the system is operating. The minimum requirement is 512GB.
Random Access Memory (RAM)	Hardware to boost system development productivity. The minimum requirement is 16GB.

3.3 Software Requirements

Table 4 below shows the software requirements for the project development process.

Table 4 *Software Requirements*

Software	Description
Visual Studio Code	Software for produces.
XAMPP	A software that is used to build a database using MySQL.
PHP	The languages that are used to develop the system. The version used is 7.25
Composer	The open-source dependency management tool for PHP. The version used is 2.6.6.
Figma	Draw wireframe
Draw.io	Draw entity relationship

4. System Analysis and Design

This section explores a systematic analysis and design of the sport rental system. The objective is to define system requirements and formulate a well-structured design.

4.1 System Requirements

Two types of requirements are required for the proposed system, which are functional requirements and non-functional requirements shown in Table 5 and Table 6.

Table 5 *Functional Requirement*

Module	Description	User
Register	The user can register an account using their name, email, phone number and password	Users
Login	The user inserts the correct email and password to log into the system	Administrator User
Configure user	The administrator can perform the CRUD function on the user.	Administrator
Configure system	The administrator can perform CRUD functions on the city and hall.	Administrator
Configure sport rental	The administrator can perform the CRUD function on sport rental.	Administrator
Configure event, facility rental	The administrator can perform CRUD functions on event, facility rental.	Administrator
View Bookings	The administrator can view the user bookings.	Administrator
Dashboard	Display the city and hall of Cameron Highland	User
Select rental	User able to select their rental type such as sport rental or event rental.	User
Select date	User should be allowed to select a date for their rental.	User
Select the timeslot and court number	User should be allowed to select timeslot and court number for their sport rental.	User
Select Facility	User able to select the facility for event rental	User
Payment	User can complete the payment after they select their rental.	User
Booking History	User can see their upcoming booking and pass bookings.	User

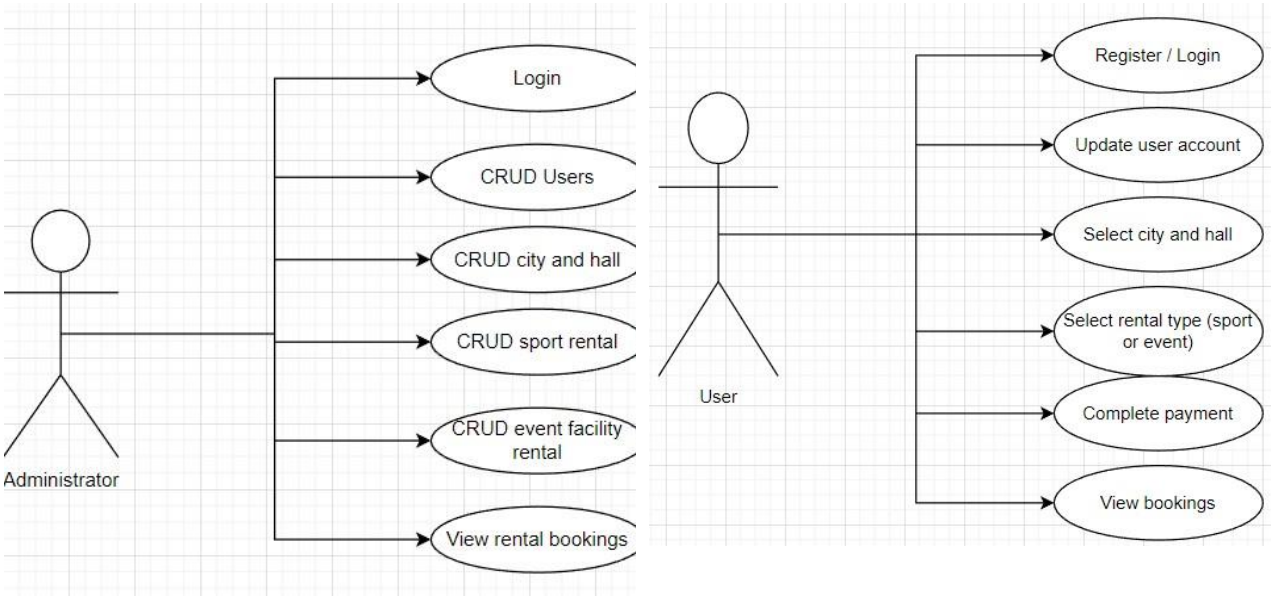
Table 6 *Non-Functional Requirement*

Module	Functionality
Operational	The system should be able to work on any web browser.
Performance	The system is available for users to use 24 hours per day.
	The system can respond to the user's interaction below 1 section.
	The system can bandwidth with many users at one time.
Security	The system should allow users to create an account with a unique email.
	The password will be hashed.
	The system should assign access privileges based on the role of users.
Usability	The system should be easy to learn for new users.
	The system should be user-friendly.

Applied Information Technology and Computer Science Vol. 5 No. 2 (2024) p. 598-615

4.2 Use-case Diagram

The use case diagram serves as a graphical depiction of the functionalities and interactions within a system. It illustrates the involvement of different actors and their interactions with the system. Figures 2(a) and 2(b) below show the use case diagram for the administrator and user of the proposed system.



(a) **(b)**

Fig. 2 *Use-case diagram (Administrator); (b) User*

4.3 Sequence Diagram

A sequence diagram is a type of interaction diagram that depicts the interactions between the objects in a system and the order in which they occur. This proposed system produces two sequence figures for administrator and user. Figure 3 shows the sequence diagram for the administrator. After the administrator logs into the system by entering the correct email and password, he can assign roles, manage user accounts, configure city and hall, configure sports, events and facilities and view rentals. Figure 4 shows the sequence diagram for the user. The user needs to enter the correct email and password before logging into the system. The user can update the account, select city and hall, rent sports, events and facilities, complete the payment, view the upcoming booking and pass bookings after they log into the system.

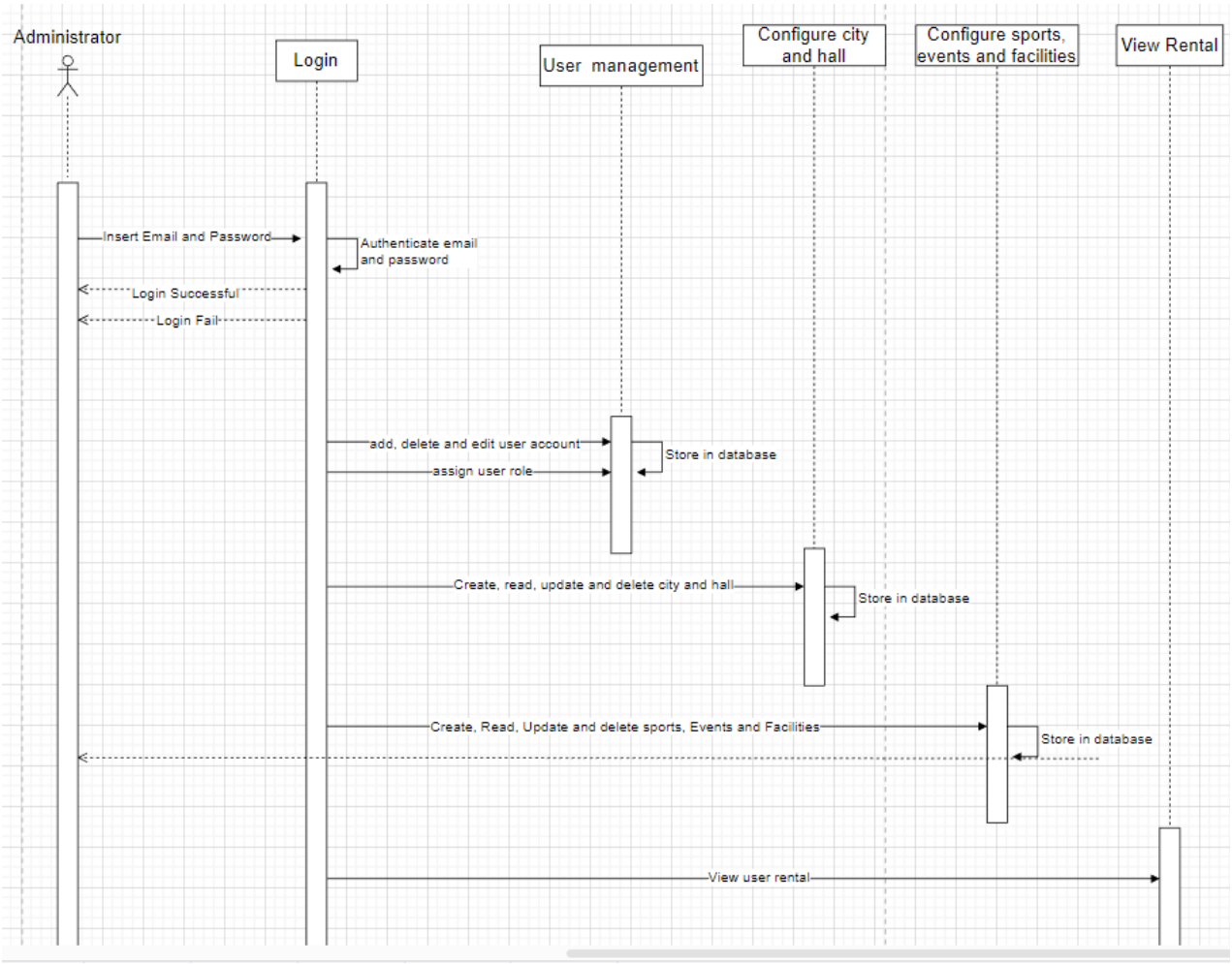


Fig. 3 *Sequence diagram for administrator*

Applied Information Technology and Computer Science Vol. 5 No. 2 (2024) p. 598-615

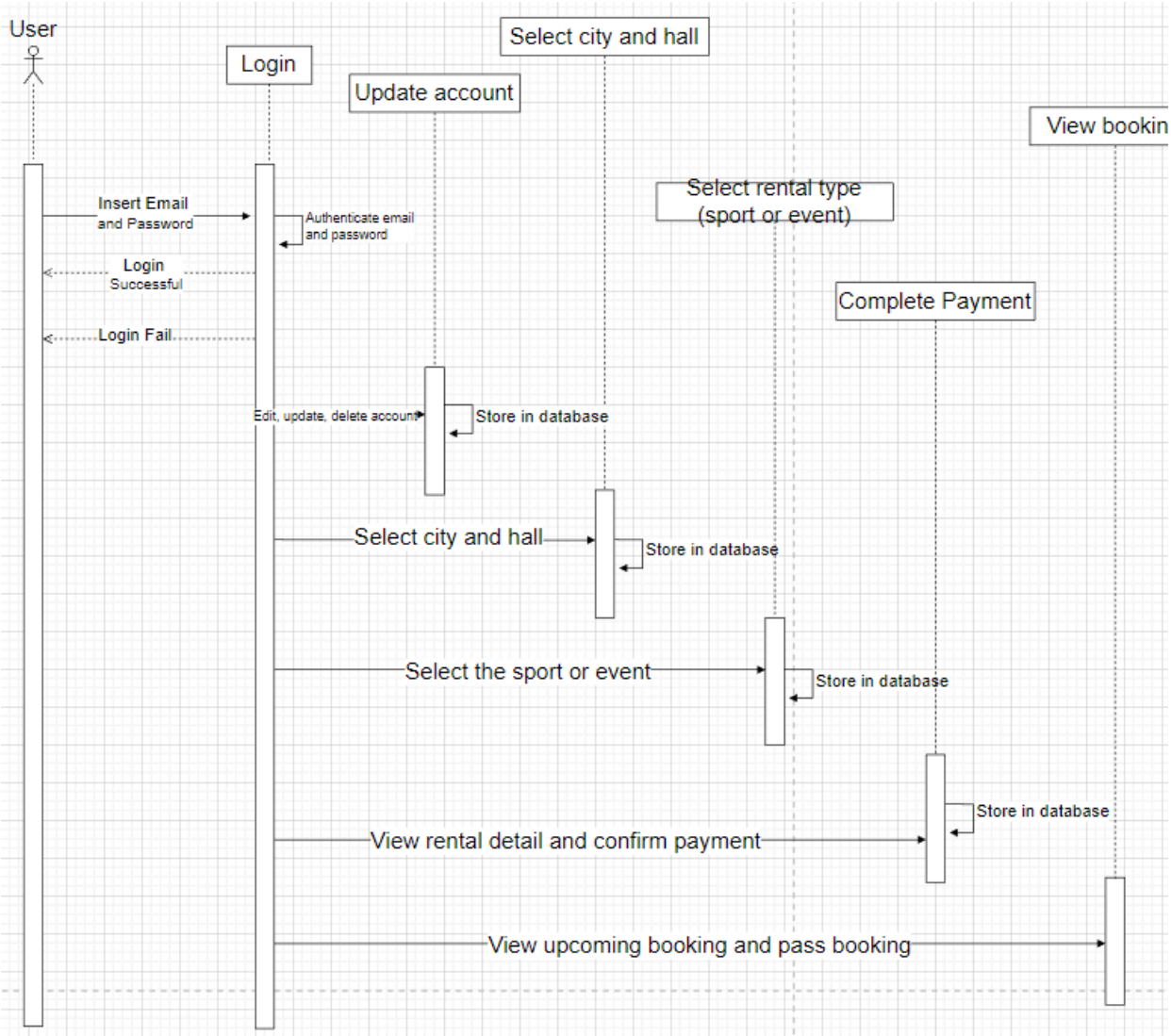


Fig.4 *Sequence diagram for user*

4.4 Flow Chart

The system flowchart shows the activity from the start to the end of the system. It will show the user flow when the user is using the system. Figure 5 below shows the system flow chart.

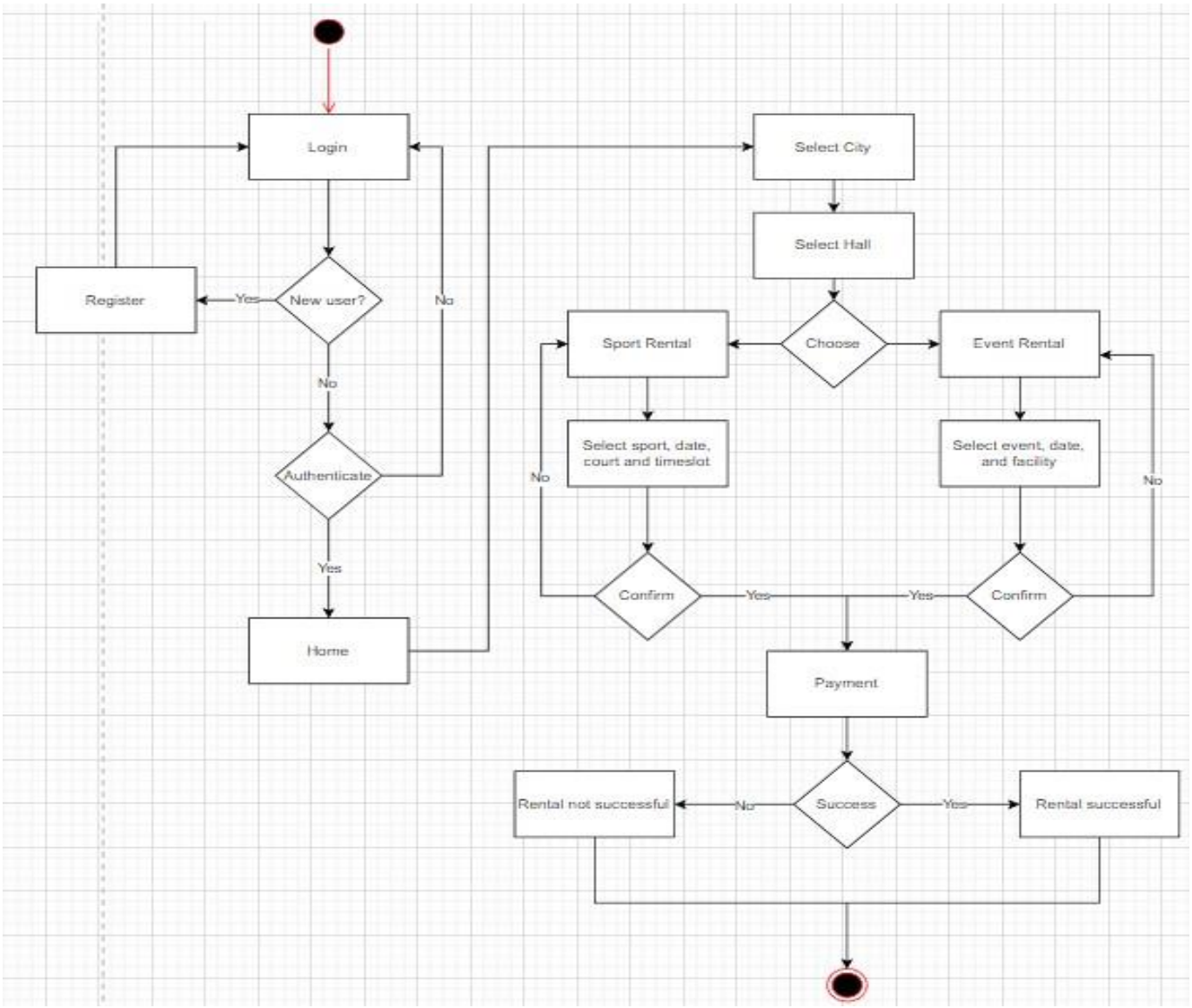


Fig.5 *Flow Chart*

4.5 Class Diagram

A class diagram is a visual representation of the relationships between classes and their members in an object-oriented program. The diagram shows the inheritance hierarchy of classes and their methods, fields and properties. The purpose of a class diagram is to document the architecture and design of an application. Figure 6 below illustrates the class diagram of the proposed rental system.

Applied Information Technology and Computer Science Vol. 5 No. 2 (2024) p. 598-615

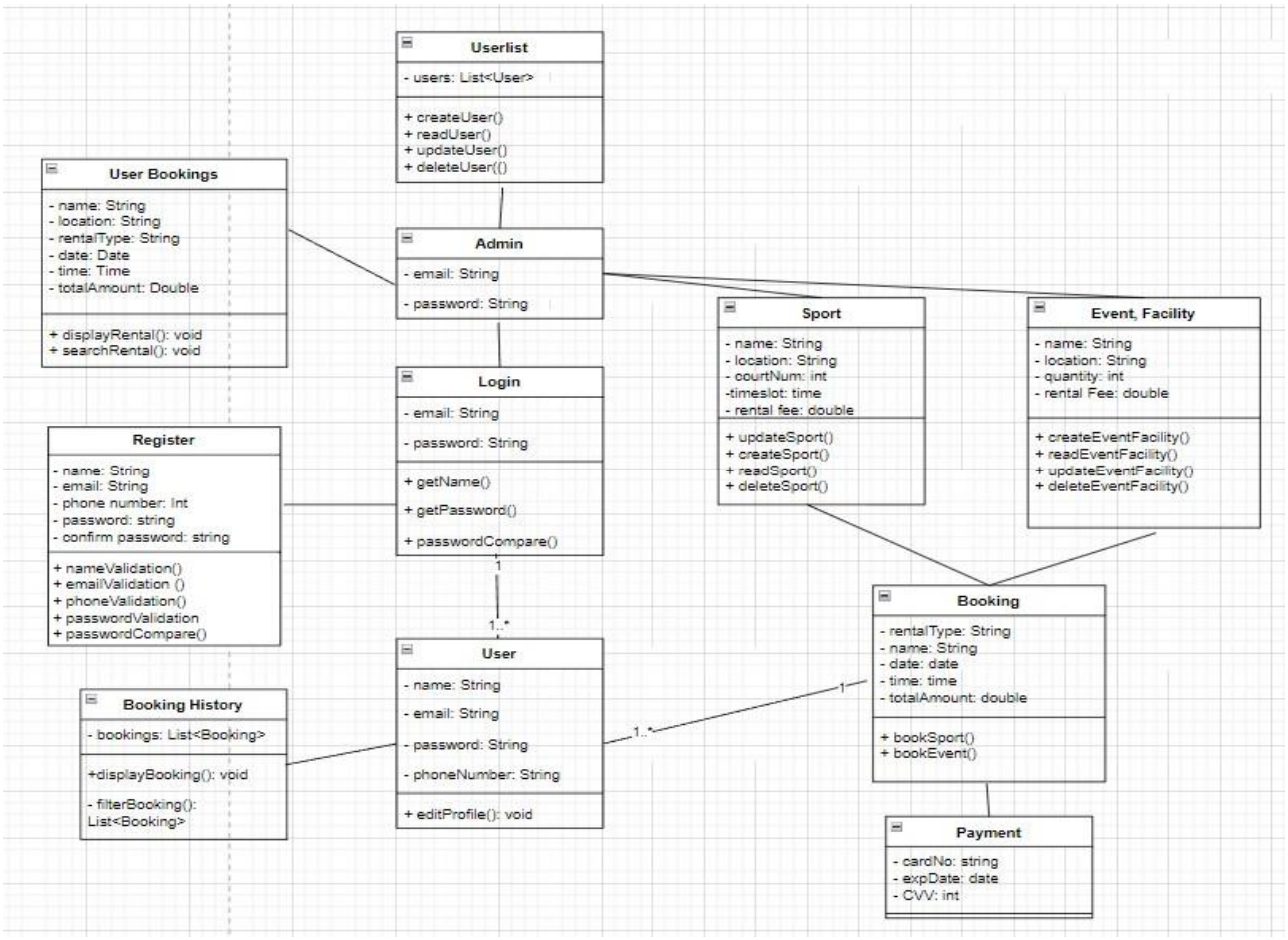


Fig. 6 *Class diagram for rental system*

4.6 General System Architecture

The system architecture is vital for organizing and managing the structure and behaviour of the proposed system. It provides a clear understanding of the system's structure. Figure 7 shows the system architecture of the proposed sports, events, and facilities rental system. When the administrator enters the correct email and password, he can log in to the system. Then he can be able to configure the hall, and city, configure the sport, event, and facility, assign roles, and CRUD user details. All the activities will be stored in the database and passed to the server. Meanwhile, when the user enters the correct email and password, he will log in successfully. For first-time users, they need to register first before they log in to the system. When the user enters the system, they can configure their account and rent sports, events or facilities. All the activities will be stored in the database and passed to the server.

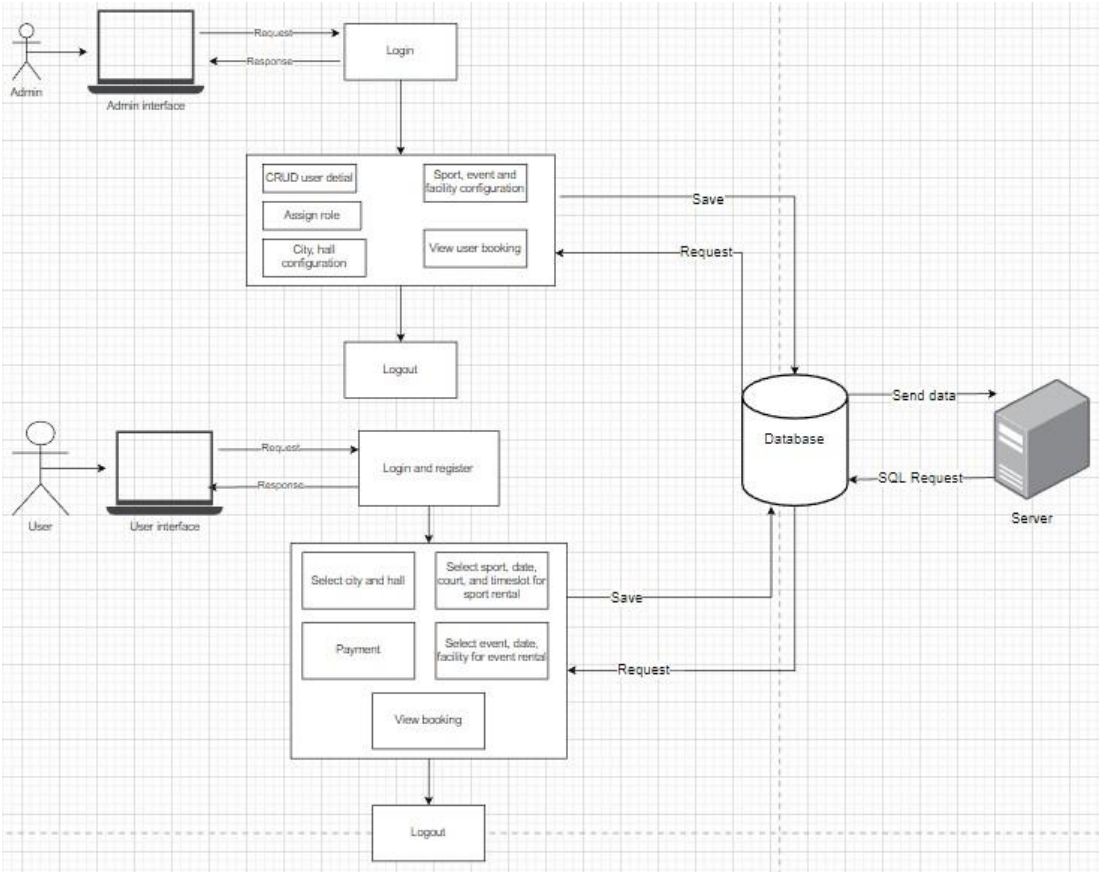


Fig.7 *System Architecture*

4.7 System Interface Design

Interface design refers to creating the visual of the proposed system that users interact with. It includes the visual design and arrangement of elements and controls to provide a user-friendly experience. Figure 8 shows the user dashboard. Figure 9 shows the administrator dashboard.

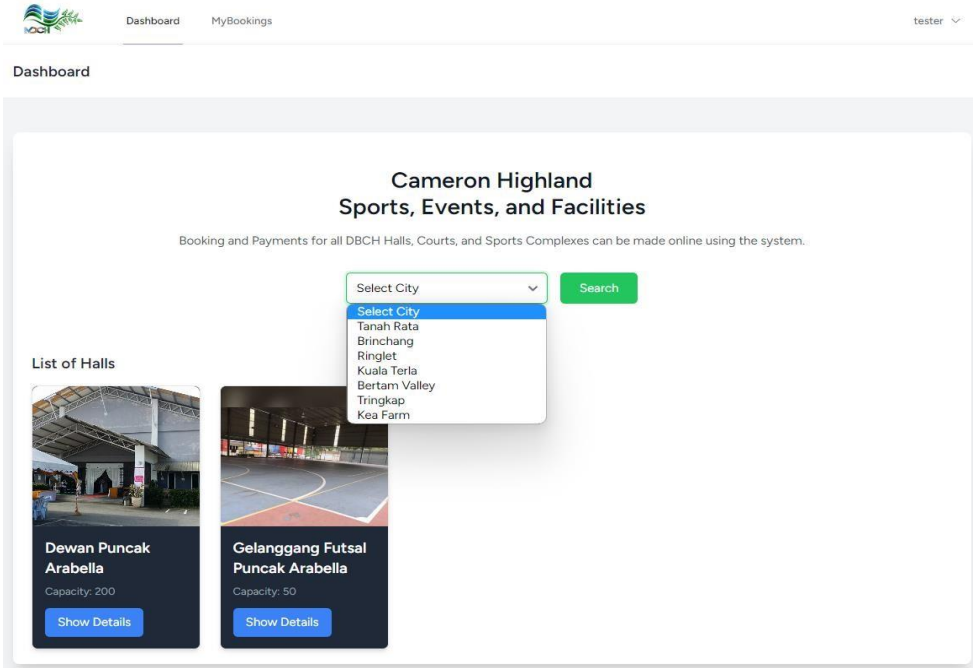


Fig.8 *User dashboard*

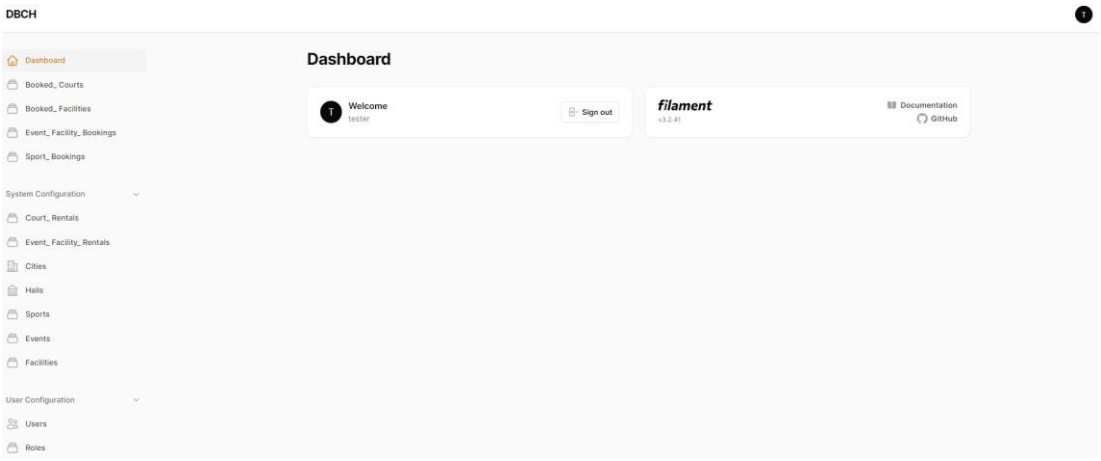


Fig.9 *Administrator Dashboard*

5. Implementation and Testing

This chapter explains the implementation and testing of the rental system. This chapter will describe the implementation of the module in this system.

5.1 Login / Sign up module

Figure 10 (a) shows the code segment for the login page. The user must insert the correct email and password to login to the system. If the user performs too many login attempts, the error message "Has too many login attempts" will show and the user needs to wait until the cooldown ends to continue login. Figure 10 (b) shows the code segment for the signup page. Users need to insert their name, email, password phone number to sign up to the system. The user needs to make sure to pass the password rules such as having at least 8 characters, having numbers, and having symbols.

```php
public function login(Request $request)
{
    $this->validateLogin($request);

    if (method_exists($this, 'hasTooManyLoginAttempts') &&
    }

    if ($this->attemptLogin($request)) {
    }

    $this->incrementLoginAttempts($request);

    return $this->sendFailedLoginResponse($request);
}

/**
 * Validate the user login request.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return void
 *
 * @throws \Illuminate\Validation\ValidationException
 */
protected function validateLogin(Request $request)
{
    $request->validate([
        $this->username() => 'required|string',
        'password' => 'required|string',
    ]);
}
```

```php
public function create(array $input): User
{
    Validator::make($input, [
        'name' => ['required', 'string', 'max:255'],
        'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
        'password' => $this->passwordRules(),
        'terms' => Jetstream::hasTermsAndPrivacyPolicyFeature() ? ['accepted', 're
        'phone' => ['required', 'string', 'min:10', 'max:11'],
    ])->validate();

    return User::create([
        'name' => $input['name'],
        'email' => $input['email'],
        'password' => Hash::make($input['password']),
        'phone' => $input['phone'],
    ]);
}
```

Fig. 10 *User (a)Login module; (b) Sign-up module*

5.2 System Configuration Module (Administrator)

The system configuration module will allow the administrator to create, read, update and delete the data of users, roles, city, hall, sports, events and facilities from the database. Figure 11 (a) shows the code segment for creating the user. Figure 11 (b) shows the code segment for deleting user data. Figure 12 (a) shows the code segment for view user. Figure 12(b) shows the code segment for updating user data.

```php
class CreateUser extends CreateRecord
{
    0 references
    protected static string $resource = UserResource::
}
```

```php
protected function getHeaderActions(): a
{
    return [
        Actions\DeleteAction::make(),
    ];
}
```

Fig. 11 *(a) Create user; (b) Delete User*

```php
public static function table(Table $table): Table
{
    return $table
        ->columns([
            Tables\Columns\TextColumn::make('name')
                ->searchable()
                ->sortable(),
            Tables\Columns\TextColumn::make('email')
                ->searchable(),
            Tables\Columns\TextColumn::make('phone')
                ->searchable(),
            Tables\Columns\TextColumn::make('email_verified_at')
                ->dateTime()
                ->sortable()
                ->toggleable(isToggledHiddenByDefault: true),
            Tables\Columns\TextColumn::make('role')
                ->searchable(),
            Tables\Columns\TextColumn::make('status'),
            Tables\Columns\TextColumn::make('two_factor_confirmed_at')
                ->dateTime()
                ->sortable()
                ->toggleable(isToggledHiddenByDefault: true),
            Tables\Columns\TextColumn::make('current_team_id')
                ->numeric()
                ->sortable()
                ->toggleable(isToggledHiddenByDefault: true),
            Tables\Columns\TextColumn::make('profile_photo_path')
                ->searchable()
                ->toggleable(isToggledHiddenByDefault: true),
            Tables\Columns\TextColumn::make('created_at')
                ->dateTime()
                ->sortable()
                ->toggleable(isToggledHiddenByDefault: true),
            Tables\Columns\TextColumn::make('updated_at')
                ->dateTime()
                ->sortable()
                ->toggleable(isToggledHiddenByDefault: true),
        ])
        ->filters([
            //
        ])
        ->actions([
            Tables\Actions\EditAction::make(),
            Tables\Actions\DeleteAction::make(),
        ])
        ->bulkActions([
            Tables\Actions\BulkActionGroup::make([
                Tables\Actions\DeleteBulkAction::make(),
            ]),
        ]);
}
```

```php
public static function form(Form $form): Form
{
    return $form
        ->schema([
            Forms\Components\TextInput::make('name')
                ->required()
                ->maxLength(255),
            Forms\Components\TextInput::make('email')
                ->email()
                ->required()
                ->maxLength(255),
            Forms\Components\TextInput::make('phone')
                ->tel()
                ->maxLength(11)
                ->default(null),

            Forms\Components\TextInput::make('password')
                ->password()
                ->dehydrateStateUsing(fn ($state) => Hash::make($state))
                ->dehydrated(fn ($state) => filled($state))
                ->required(fn (string $context): bool => $context === 'create'),

            Forms\Components\Select::make('status')
                ->options([
                    'active' => 'Active',
                    'not_active' => 'Not Active',

                ])
                ->required(),

            Forms\Components\Select::make('role')
                ->preload()
                ->options(function () {
                    return Role::pluck('name', 'name')->toArray();
                })
                ->required(),
        ]);
}
```

Fig. 12 *Code Segment(a) View User; (b) Update user*

5.3 Sport Rental Module

For this session, the user must select the city, hall, sport, date, timeslot and payment to successfully rent the desired sport. Figure 13 (a) shows the code segment select city code. Figure 13 (b) shows the code segment for the select hall code. Figure 14 and Figure 15 show the code segment for selecting sport and date codes. Then figure 16 shows the timeslot code. The payment code segment is shown in Figure 17.

```php
public function render()
{
    $cities = City::all();
    return view('livewire.city-selection-component', compact('
}

0 references | 0 overrides
public function searchCities()
{
    // Fetch halls related to the selected city
    $halls = Halls::where('city_id', $this->selectedCity)->get(

    // Emit an event to handle redirection in the parent compo
    $this->emit('searchResults', $halls);

}
```

```php
public function search()
{
    $halls = Halls::query(); // Get the list of halls

    // Apply search filter
    if (!empty($this->selectedCity)) {
        $halls = $halls->where('cities_id', $this->selecte
    }
    $this->halls = $halls->get();
    // return $halls;
}
```

Fig. 13 *Code Segment(a) Select City; (b) Select Hall*

```php
public function selectSport($sportId)
{
    $this->selectedSport = Sports::with('court_rentals')->find($sportId);
    $court_rental = $this->selectedSport->court_rentals->where('halls_id', $this->selectedHall)->first();
    if ($court_rental) {
        $this->numberOfCourts = $court_rental->number_of_court;
        $this->fetchAvailableTimeSlots();
        $this->dispatch('sportSelected', $this->numberOfCourts, $this->selectedHall, $this->selectedSport->id);
    }
}
```

Fig. 14 *Select Sport code segment*

```php
public function mount($selectedHall)
{
    $this->selectedHall = $selectedHall;
    $this->sports = Court_Rental::where('halls_id', $this->selectedHall)->with('sports')->get();
    $this->date = now()->format('Y-m-d');
}
```

Fig. 15 *Select Date code segment*

```php
public function fetchAvailableTimeSlots()
{
    $this->selectedTimes = [];
    $startTime = Carbon::createFromTime(9, 0, 0, 'Asia/Singapore');
    $endTime = Carbon::createFromTime(23, 0, 0, 'Asia/Singapore');
    $interval = CarbonInterval::hour();
    $now = Carbon::now('Asia/Singapore');
    $thirtyDaysLater = $now->copy()->addDays(30);

    $courtRentalId = $this->selectedSport->court_rentals->where('halls_id', $this->selectedHall)->first()->id;

    for ($current = $startTime; $current->lessThan($endTime); $current->add($interval)) {
        foreach (range(1, $this->numberOfCourts) as $courtNumber) {
            $courtName = 'Court_' . $courtNumber;
            $timeSlot = TimeSlot::where('date', $this->date)
                ->where('time', $current->format('H:i:s'))
                ->where('court_rental_id', $courtRentalId)
                ->where('court_number', $courtNumber)
                ->first();

            $booking = Sport_Bookings::where('halls_id', $this->selectedHall)
                ->where('sports_id', $this->selectedSport->id)
                ->where('use_date', $this->date)
                ->where('confirmed', 1)
                ->whereHas('bookedCourts', function ($query) use ($current, $courtNumber) {
                    $query->where('start_time', $current->format('H:i:s'))
                        ->where('court_number', $courtNumber);
                })
                ->first();

            $isPast = $now->greaterThanOrEqualTo(Carbon::parse($this->date . ' ' . $current->format('H:i:s')));
            $isBeyondThirtyDays = Carbon::parse($this->date)->greaterThan($thirtyDaysLater);

            $this->selectedTimes[$courtName][] = [
                'time' => $current->format('H:i') . ' - ' . $current->copy()->addHour()->format('H:i'),
                'available' => !$isPast && !$isBeyondThirtyDays && ($timeSlot ? $timeSlot->available : true) && !$booking,
                'selected' => false,
            ];
        }
    }
}
```

Fig. 16 *Timeslot code segment*

```php
0 references | 0 overrides
public function submitPayment()
{
    // Convert month name to number
    $monthNames = [
        'January' => '01', 'February' => '02', 'March' => '03', 'April' => '04',
        'May' => '05', 'June' => '06', 'July' => '07', 'August' => '08',
        'September' => '09', 'October' => '10', 'November' => '11', 'December' => '12'
    ];
    $this->expMonth = $monthNames[$this->expMonth] ?? null;
    $validatedData = $this->validate([
        'cardName' => 'required|string|max:255',
        'cardNum' => 'required|string|regex:/^[0-9\s]{16,19}$/',
        'expMonth' => ['required', 'string', 'size:2', 'regex:/^(0[1-9]|1[0-2])$/'],
        'expYear' => 'required|integer|min:' . now()->year,
        'cvv' => 'required|string|digits:3',
    ], [
        'cardNum.regex' => 'The credit card number format is invalid.',
        'expMonth.size' => 'The expiration month must be 2 characters.',
        'expMonth.regex' => 'The expiration month format is invalid.',
        'expYear.min' => 'The expiration year must be a valid future year.',
    ]);
    DB::beginTransaction();
    try {
        // Check if booking_id exists in rentalDetails
        if (!isset($this->rentalDetails['booking_id'])) {
            throw new \Exception('Booking ID is missing from rental details');
        }

        $sportBooking = Sport_Bookings::find($this->rentalDetails['booking_id']);
        if ($sportBooking) {
            // Update the booking status to confirmed only if payment is successful
            $sportBooking->confirmed = 1;
            $sportBooking->save();
            Log::info('Booking confirmed', ['booking_id' => $sportBooking->id]);

            // Retrieve booked courts for the confirmed booking
            $bookedCourts = booked_courts::where('sport_bookings_id', $sportBooking->id)->get();

            // Update the corresponding time slots to not available only if the booking is confirmed
            foreach ($bookedCourts as $bookedCourt) {
                TimeSlot::where('date', $sportBooking->use_date)
                    ->where('time', $bookedCourt->start_time)
                    ->where('court_rental_id', $sportBooking->court_rental_id)
                    ->where('court_number', $bookedCourt->court_number)
                    ->update(['available' => false]);
            }
        }
        DB::commit();
        // Set success message
        session()->flash('success', 'Payment completed successfully!');
    } catch (\Exception $e) {
        DB::rollBack();
        Log::error('Payment processing error: ' . $e->getMessage());
        session()->flash('error', 'Payment failed. Please try again.');
    }
}
```

Fig. 16 *Payment code segment*

6. Result and Discussion

This section presents the results of the proposed system that are discussed. There are two types of testing results for the proposed system. The testing phase is performed on a whole system to verify the security and applicable requirements for the project.

6.1 Test Plan Result

The results of the test plan are presented here. Table 7 displays the result of the functionality test plan for the user, Table 8 displays the results of the functionality test plan for the administrator and Table 9 outlines the results of the security test plan.

Table 7 *Functionality Test Plan Result for User*

Module	Test Plan	Expected Result	Result
Registration	The user enters data incompletely	The error message "Please fill out this field" will show	Pass
	The password and confirmed password sections do not match.	The error message "Password not same" will display.	Pass
	The user enters data successfully.	The success message "register successfully" will display.	Pass
Login	The user enters data incompletely.	The error message "Please fill out this field" will display.	Pass
	The user enters the incorrect email.	The error message "Invalid email or password" will display.	Pass
	The user enters an incorrect password.	The error message "Invalid email or password" will display.	Pass

Table 7: (Cont)

	The user enters the correct email and password.	Login successfully and redirect to the dashboard.	Pass
Dashboard	The user can select the city.	The user is redirected to the hall page.	Pass
Hall page	The user can select to rent the sport or event.	The user is redirected to the sport rental page or event rental page.	Pass
Sport rental page	The user can select the desired sport, date, court and timeslot.	The user can successfully select the desired sport, date, court, and timeslot.	Pass
Event rental page	The user can select the desired event, date, and facility.	The user can choose the desired event, date and facility successfully.	Pass
Payment	The user is redirected to this page after the rental and complete payment.	The user is redirected to this page after the rental and complete payment successfully.	Pass
Booking History	The user can check upcoming bookings and past bookings.	The user can check upcoming bookings and past bookings successfully.	Pass

Table 8 *Functionality Test Plan Result for Administrator*

Module	Test Plan	Expected Result	Result
Login	The administrator enters data incompletely.	The error message "Please fill out this field" will display.	Pass
	The administrator enters the incorrect email.	The error message "Invalid email or password" will display.	Pass
	The administrator enters an incorrect password.	The error message "Invalid email or password" will display.	Pass
	The administrator enters the correct email and password.	Login successfully and redirect to the dashboard.	Pass
Configure user	The administrator can perform the CRUD function on the user.	The administrator can perform the CRUD function on the user successfully.	Pass
Configure system	The administrator can perform the CRUD function in the city, hall.	The administrator can perform the CRUD function on the city, hall successfully.	Pass
Configure Sport Rental	The administrator can perform the CRUD function on the sport rental information.	The administrator can perform the CRUD function on the sport rental information successfully.	Pass
Configure Event, Facility Rental	The admin can perform the CRUD function on the event, and facility rental information.	The admin can perform CRUD function on the event, and facility rental information successfully.	Pass
View User rental	The administrator can view the rental details of the user.	The administrator can view the rental details of the user successfully.	Pass

Table 9 *Security Test Plan Result*

No	Check List	Result
1	Ensure the error message does not directly indicate which part of the authentication data is incorrect. For example, an error message should not show "incorrect email" or "incorrect password".	Pass
2	Enforce the complexity of the password. For example, using alphabetic and numeric as well as special characters to create passwords.	Pass
3	Enforce the password length which is a minimum of eight characters.	Pass
4	Password should be obscured in the textbox.	Pass
5	Restrict user access privilege based on role.	Pass
6	Prevent SQL injection into the system.	Pass

6.2 User Acceptance Form

This section will show the result of the user acceptance testing. The purpose of conducting this test is to collect the user experience from the user side when using the system. The test is separated into two parts which are user acceptance testing and administrator acceptance testing. The figure below shows the result of user and administrator acceptance testing of the system.

Table 10 *User Acceptance Form Result*

Test Plan	Ranking					Total
	1	2	3	4	5	25
User can register account.					25	25
User can login into the system.					25	25
User can update their information.					25	25
User can select the city.					25	25
User can select the hall.					25	25
User can select the rental type (sport or event).					25	25
User can select date and facilities for event rental.					25	25
User can select the date, court number and time slot for sport rental.					25	25
User able to complete payment successfully.					25	25
User able to view booking history.				1	24	25

Table 11 *Administrator Acceptance Form Result*

No	Question	Result
1	The administrator can log into the system.	Pass
2	The administrator can perform the CRUD function on the user.	Pass
3	The administrators can perform CRUD functions in the city, hall.	Pass
4	The administrator can perform CRUD functions on sports rental information.	Pass
5	The administrator can perform the CRUD function on event, and facility rental information.	Pass
6	The administrator can view the rental details of the users.	Pass

7. Conclusion

In conclusion, the proposed Sports, Events and Facilities Rental System has been developed with complete functionality. The system successfully fulfils the objectives defined in the project scope, system requirements and user requirements. The Sports, Events, and Facilities Rental System offers a few advantages which are streamlining the user rental process and administrator management system process. However, there are also consists of disadvantages to the system which are the user cannot perform actual payment process and the system's lack of email verification. Based on the analysis of the system's advantages and disadvantages, there are the following improvements that can be made to the system in future. The first is to implement the payment gateway into the system. This can allow users to perform actual payment processes through the system. This can

increase the user experience. Secondly, implement email verification. This is to confirm the identity of users and prevent the creation of fake accounts.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, University Tun Hussein Onn Malaysia for its support.

Conflict of Interest

The authors declare that there is no conflict of interest regarding the publication of the paper.

Author Contribution

The author confirms sole responsibility for the following: study conception and design, data collection, analysis and interpretation of results, and manuscript preparation.

References

[1] M. Nizam Osman, N. Md Zain, Z. Paidi, K. Anwar Sedek, and M. NajmuddinYusoff, "Article 6 Online Car Rental System using Web-Based and SMS Technology," 2017. [Online]. Available: https://crinn.conferencehunter.com

[2] L. Potwarka and R. Snelgrove, "Managing Sport Events for Beneficial Outcomes: Theoretical and Practical Insights," *Event Management*, vol. 21, Dec. 2017, doi: 10.3727/152599517X14878772869522.

[3] A. Fernández-Martínez, J. A. Tamayo-Fajardo, R. Nuviala, D. Cabello-Manrique, and A. Nuviala, "The management of major sporting events as an antecedent to having the city recommended," *Journal of Destination Marketing & Management*, vol. 19, p. 100528, 2021, doi: https://doi.org/10.1016/j.jdmm.2020.100528.

[4] M. Stobart Simon and Vassileiou, "Introduction to PHP," in *PHP and MySQL Manual: Simple, yet Powerful Web Programming*, London: Springer London, 2004, pp. 7–11. doi: 10.1007/978-0-85729-404-3_2.

[5] M. I. Kausar Bagwan and P. D. Swati Ghule, "A Modern Review on Laravel-PHP Framework," 2019.

[6] M. Amini *et al.*, "MAHAMGOSTAR.COM AS A CASE STUDY FOR ADOPTION OF LARAVEL FRAMEWORK AS THE BEST PROGRAMMING TOOLS FOR PHP BASED WEB DEVELOPMENT FOR SMALL AND MEDIUM ENTERPRISES," 2020. [Online]. Available: https://www.mahamgostar.com/

[7] A. Sunardi and Suharjito, "MVC architecture: A comparative study between laravel framework and slim framework in freelancer project monitoring system web based," in *Procedia Computer Science*, Elsevier B.V., 2019, pp. 134–141. doi: 10.1016/j.procs.2019.08.150.

[8] Pitchbooking.com, 2023. https://pitchbooking.com/solution

[9] "Spacebase | Rent inspiring meeting rooms and event spaces," *www.spacebase.com*. https://www.spacebase.com/en/

[10] "About Us," *Sports-Booker.com*. https://sports-booker.com/discover/about-us/

[11] K. Risener, "A Study of Software Development Methodologies." [Online]. Available: https://scholarworks.uark.edu/csceuht/103

[12] "Agile project management with Scrum."