**4. Implementation**

This chapter provides a comprehensive overview of how the major features of the *Stayvista System* were implemented. The system adopts a modern architecture leveraging PHP for server-side logic, Supabase as Backend-as-a-Service (BaaS), and contemporary web technologies for an optimal user experience.

**4.1 System Architecture and Technology Stack**

**4.1.1 Architectural Overview**

The system follows a service-oriented architecture pattern where PHP serves as the presentation and business logic layer. At the same time, Supabase provides comprehensive backend services, including authentication, database operations, and file storage. This approach minimises custom backend complexity while ensuring scalability and security.

**Core Components:**

**Frontend Layer**: Server-rendered PHP pages with Tailwind CSS styling and JavaScript for dynamic interactions

**Business Logic Layer**: PHP service classes encapsulating remote API calls

**Backend Services**: Supabase ecosystem (Auth, PostgREST, Storage)

**Database Layer**: PostgreSQL managed by Supabase with Row Level Security (RLS)

**4.1.2 Technology Justification**

The technology stack was selected based on the following criteria:

**PHP**: Server-side rendering for better SEO and initial page load performance

**Supabase**: Reduces infrastructure overhead while providing enterprise-grade security

**Tailwind CSS**: Utility-first approach for rapid UI development

**PostgreSQL**: Robust relational database with advanced features like RLS

**4.2 User Authentication and Role-Based Access Control**

**4.2.1 Registration System Implementation**

The user registration system implements a comprehensive flow supporting multiple user types (students, landlords, administrators) with email verification capabilities.



**Key Features:**

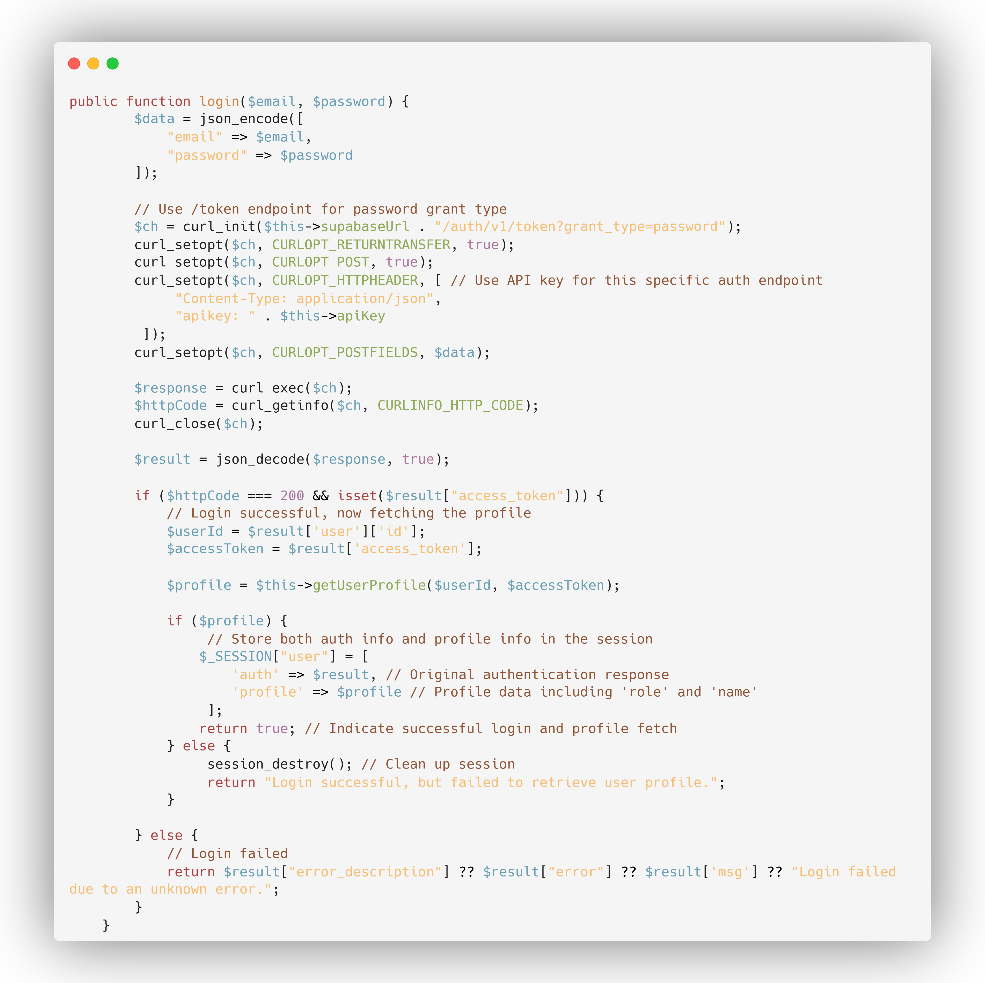
* Email/password authentication with optional verification
* Automatic profile creation with role assignment
* Unique identifier generation for user-friendly references
* Input validation and sanitisation

Show Image

The registration process validates user inputs, creates authentication records via Supabase Auth, and establishes user profiles with appropriate role assignments. The system generates unique 5-digit identifiers for improved user experience.

**4.2.2 Authentication Flow and Session Management**

Login functionality authenticates users against Supabase and establishes secure server-side sessions containing both authentication tokens and user profile data.



**Security Measures:**

* Server-side session storage
* Token-based authentication with Supabase
* Role-based page access controls
* Automatic session clean-up and timeout handling

**4.2.3 Role-Based Access Control (RBAC)**

The system implements granular access controls based on user roles:

* **Students**: Browse listings, view details, submit reports
* **Landlords**: CRUD operations on properties, dashboard access, profile management
* **Administrators**: User management, content moderation, system oversight



**4.3 Property Management System**

**4.3.1 Property Creation and Listing Management**

Landlords can create comprehensive property listings through a multi-step form process that handles property details, image uploads, and amenity selections.

**Implementation Features:**

* Multi-part form(different parts: property details, images and amenities) with validation
* Image upload with file type and size restrictions
* Geographic coordinate integration for mapping using …………….
* An amenity association through many-to-many relationships

Property details upload:



Amenities association:



**4.3.2 Image Handling and Storage Architecture**

The system implements a sophisticated image management system using Supabase Storage with user-specific folder structures and security policies.

**Image Processing Pipeline:**

1. Client-side validation (file type, size)
2. Server-side security checks
3. Unique filename generation
4. User-specific folder organisation
5. Storage upload with authentication
6. Database reference creation



**4.3.3 Property Modification and Deletion**

Property owners can modify their listings through dedicated edit interfaces with real-time validation and preview capabilities.

**Edit Functionality:**

* Pre-populated forms with existing data
* Amenity re-synchronisation on updates
* Image management (add/remove)
* Ownership verification before modifications

**4.4 Property Discovery and Search System**

**4.4.1 Listing Display with Advanced Filtering**

The property browsing system provides comprehensive filtering capabilities, allowing students to discover relevant accommodations efficiently.

**Filter Categories:**

* University proximity targeting
* Price range specifications
* Amenity-based filtering



**4.4.2 Detailed Property Views**

Individual property pages present comprehensive information, including images, amenities, location mapping, and landlord contact details.

**Detail Page Features:**

* Image gallery with primary image highlighting
* Interactive maps using Leaflet integration
* Landlord verification status display
* Contact information with privacy controls
* Amenity listing with visual indicators

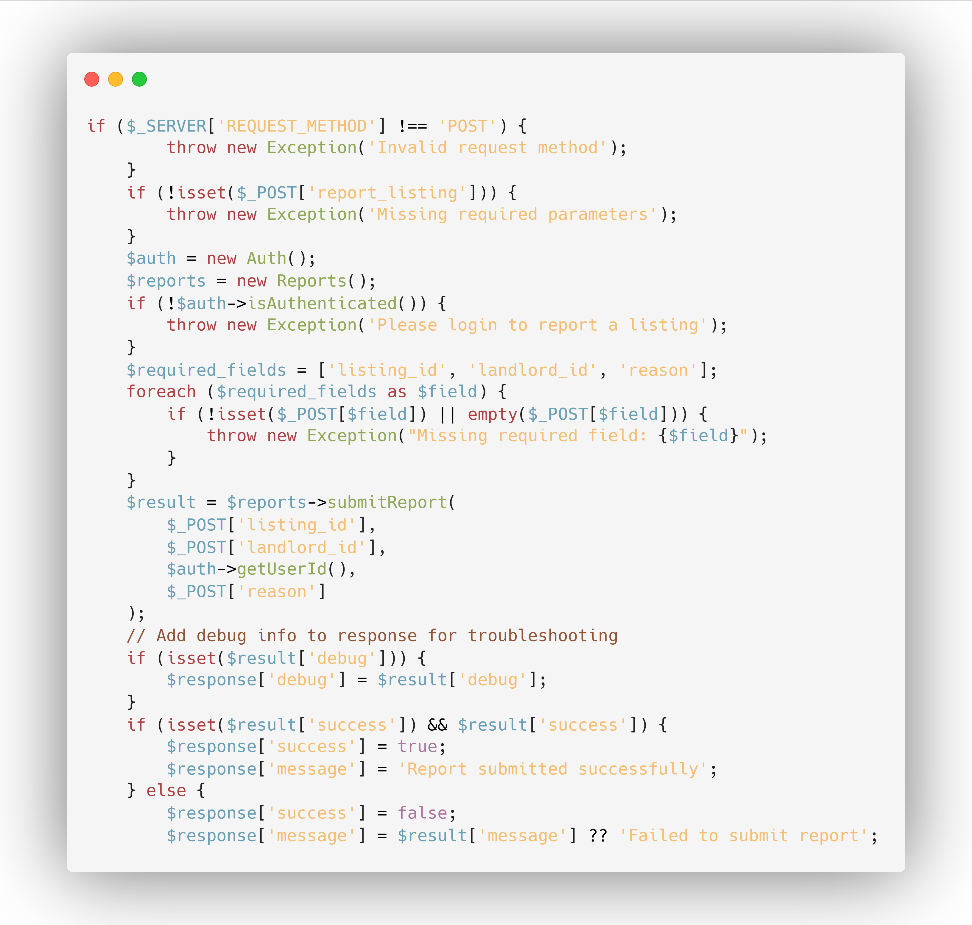
**4.5 Reporting and Moderation System**

**4.5.1 Report Submission Interface**

Students can report problematic listings through an accessible modal interface that submits reports asynchronously without page refresh.

**Reporting Features:**

* Modal-based user interface
* Asynchronous form submission
* Categorised reporting reasons
* User authentication verification
* Duplicate report prevention



**4.5.2 Administrative Moderation Dashboard**

Administrators access comprehensive tools for managing reports, user verifications, and system oversight through a dedicated dashboard.

**Administrative Capabilities:**

* Report queue management with status tracking
* User verification processing with document review
* Property moderation and removal tools
* User account management and role modifications

**4.6 Database Design and Data Management**

**4.6.1 Entity Relationship Model**

The system implements a normalised database schema supporting complex relationships between users, properties, amenities, and administrative data.

**Core Entities:**

1. **Profiles**: User account information and role assignments
2. **Properties**: Stayvista with geographic and pricing data
3. **Property Images**: File references with primary image designations
4. **Amenities**: Standardised feature catalogue
5. **Property Amenities**: Many-to-many relationship mapping
6. **Reports**: Moderation and feedback system data

// REFER TO ERD!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

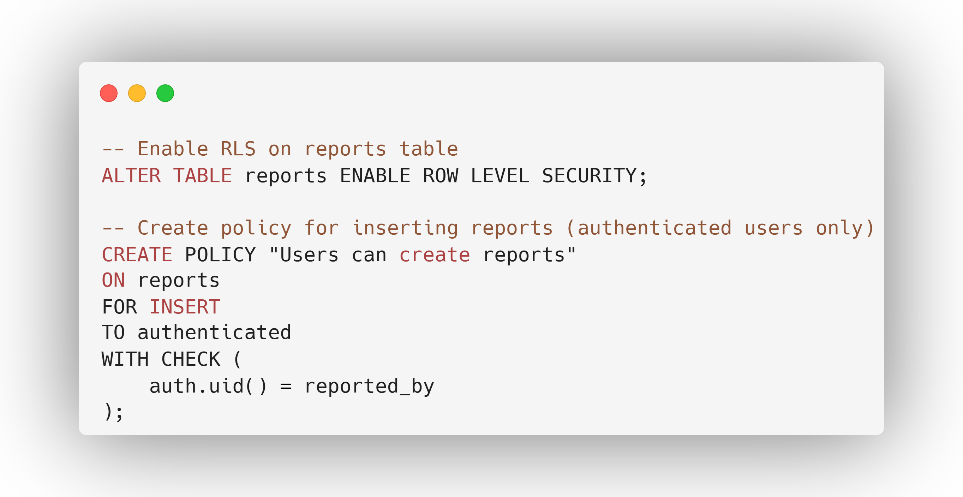
**4.6.2 Row Level Security Implementation**

Supabase RLS policies ensure data security by enforcing access controls at the database level, preventing unauthorised data access regardless of application-layer vulnerabilities.

**Security Policies:**

* Property ownership enforcement for modifications
* User profile privacy controls
* Administrative privilege verification
* Report anonymity and access restrictions

An example of a Role Level Security (RLS) policy:



**4.7 User Interface and Experience Design**

**4.7.1 Responsive Design Implementation**

The system employs Tailwind CSS utility classes to create responsive interfaces that adapt seamlessly across desktop, tablet, and mobile devices.

**Design Principles:**

* Mobile-first responsive approach
* Consistent component library
* Accessible navigation patterns
* Progressive enhancement strategies

**4.7.2 Interactive Elements and Dynamic Features**

JavaScript enhances user interactions through dynamic form handling, modal interfaces, and asynchronous operations without full-page refreshes.

**Interactive Features:**

* Real-time form validation
* Image upload progress indicators
* Modal-based interfaces for secondary actions

**4.8 API Layer and Service Architecture**

**4.8.1 Service Class Organisation**

The system organises business logic into dedicated service classes that encapsulate Supabase API interactions and provide clean interfaces for page controllers.

**Service Classes:**

* **Auth Service**: Authentication and user management
* **PropertyListings Service**: Property CRUD and retrieval operations
* **Reports Service**: Reporting and moderation functionality

Example of the PropertyListings service class:



**4.9 Security Implementation and Best Practices**

**4.9.1 Input Validation and Output Encoding**

The application adopts a defence-in-depth approach to data handling, combining client-side

usability checks with server-side validation and strict output encoding to prevent injection

and cross-site scripting.

**Security Measures:**

* **Input Validation and Output Encoding:** The application checks all data submitted by users to ensure it's safe and in the correct format. When displaying data, it is encoded so that it is treated as plain text, which prevents malicious code from running in a user's browser (Cross-Site Scripting).
* **SQL Injection Prevention:** The application avoids directly building SQL queries with user input3. Instead, it uses a secure interface (Supabase PostgREST) that separates user data from database commands, eliminating the risk of SQL injection attacks
* **File upload security:** For file uploads, the system strictly controls the file types (e.g., images, PDFs) and sizes that are allowed. Uploaded files are given unique names and stored in secure, separate locations to prevent unauthorized access or conflicts
* **CSRF protection:** Currently, forms rely on users being authenticated. However, it is recommended to add anti-CSRF tokens to all forms that change data (like creating or deleting a listing). This would add another layer of security to verify that actions are intentionally made by the user.

**4.9.2 User Authentication and Access Control**

The system implements multiple layers of security to protect user accounts and sensitive data.

**Security Features:**

* **Secure password handling:** The application does not handle or store user passwords10. This responsibility is delegated to a secure service (Supabase Auth) that uses industry-standard hashing to protect credentials
* **Session management:** After a user logs-in, a secure session is created on the server that contains their user profile and role. This session is used to control what the user can see and do, and it is destroyed upon logout to prevent misuse.
* **Token-based API:** Actions that create or update data require a secure access token from the authenticated user. This ensures every change is tied to a specific user and is subject to their permissions at the database level.
* **Role-based access control:** Security is enforced at multiple levels to ensure users only access what they're supposed to:
  + ***Pages*:** Certain pages, like the admin dashboard, can only be accessed by users with the correct role (e.g., 'administrator').
  + ***Data*:** Security rules are also applied directly at the database level (Row Level Security), so even if a user attempts to bypass the website's interface, they cannot access or modify data they don't have permission for.
  + ***User Interface*:** The website's navigation and buttons change based on the user's role, hiding options they are not authorized to use.

Together, these measures provide layered security: output encoding prevents XSS in the

presentation layer; PostgREST plus URL-encoded parameters and RLS mitigate injection

and unauthorized data access; authenticated uploads and inserts bind data mutations to the

user’s identity; and role checks at both the page and database levels ensure least-privilege

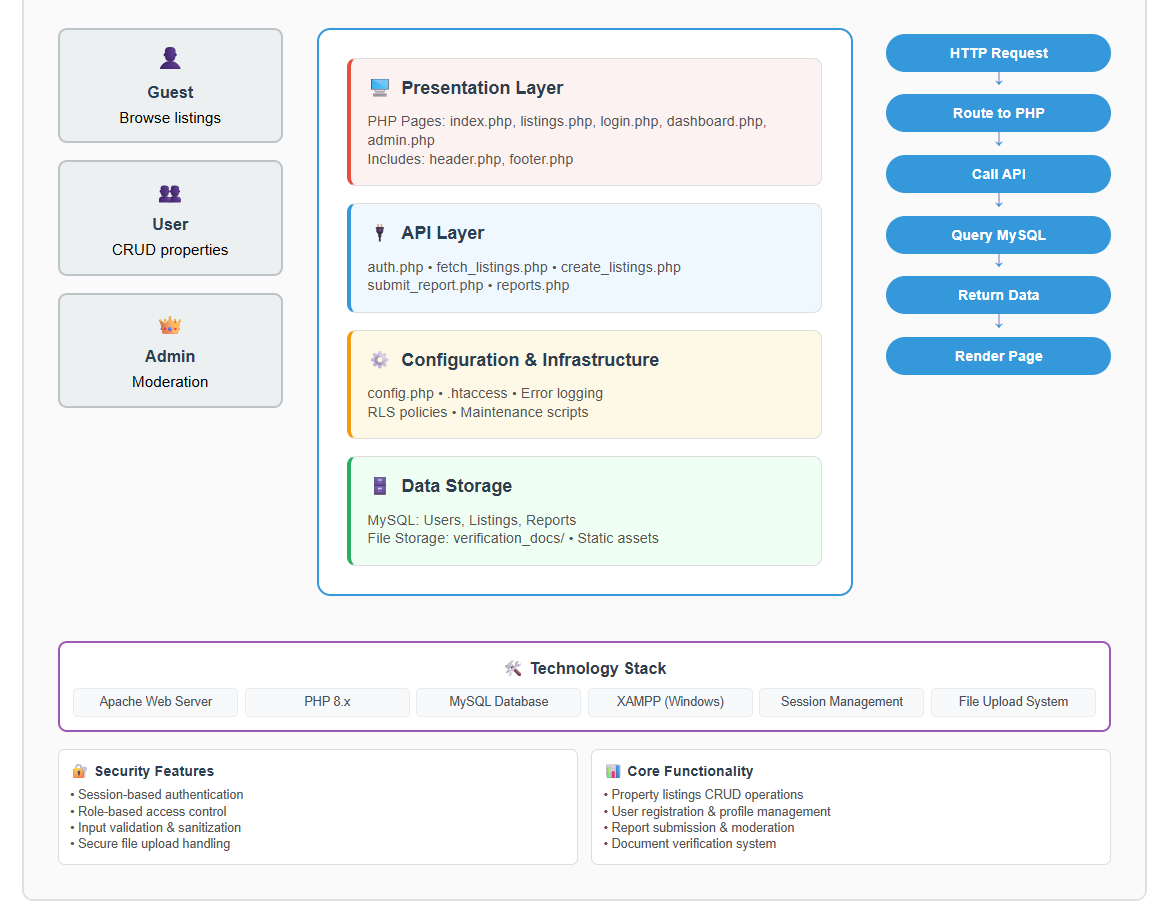
access throughout the system’s workflows.

**4.12 Summary**

The Stayvista System successfully implements a comprehensive marketplace platform that addresses the core requirements of accommodation discovery and landlord property management. The implementation demonstrates:

* **Robust Architecture**: Service-oriented design with clear separation of concerns
* **Security Focus**: Multiple layers of protection for user data and system integrity
* **User Experience**: Intuitive interfaces with responsive design and accessibility considerations
* **Maintainability**: Clean code organisation with comprehensive documentation

The system leverages modern web technologies and cloud services to deliver a production-ready platform that can be extended with additional features such as messaging, reviews, and advanced search capabilities. The implementation provides a solid foundation for continued development and scaling to meet growing user demands.

**

The successful implementation validates the architectural decisions and demonstrates the effectiveness of combining traditional server-side rendering with modern backend services to create a robust, scalable web application.