CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT317-G7 PROJECT MANAGEMENT

ENTITY RELATIONSHIP DIAGRAM (ERD)

Project Title: CampusLink

Prepared By: Josh Joshua L. Meñez / CampusLink / Developer

Date of Submission: 12/06/2025

Version: 5

Table of Contents

1. Introduction

The CampusLink database serves as the central data repository for the university opportunity management platform, facilitating connections between students and campus organizations. Its primary purpose is to:

- Centralize Opportunity Management: Store and organize all campus opportunities including assistantships, volunteer work, leadership roles, and sports tryouts in a single accessible location.
- Enable User Management: Maintain comprehensive user profiles for students, organizations, and administrators with appropriate role-based access controls.
- Support Application Processing: Track student applications to opportunities, enabling organizations to review, accept, or reject candidates efficiently.
- Facilitate Organization Verification: Manage the verification process for campus organizations, ensuring only legitimate groups can post opportunities.
- Enable Real-time Communication: Support notification systems and messaging between students and organizations.

Scope

The database encompasses all core functionalities required for the CampusLink platform:

User Management

- Student profiles with academic and skill information
- Organization profiles with contact details and verification status
- Administrator accounts with moderation capabilities
- Authentication and session management

Opportunity Management

- Creation, modification, and deletion of opportunity postings
- Categorization of opportunities by type (assistantship, volunteer, etc.)
- Deadline tracking and status management
- Location and requirement specifications

Application Processing

- Student application submissions with cover letters and resumes
- Application status tracking (submitted, reviewed, accepted, rejected)
- Communication history between applicants and organizations

Verification Systems

- Organization verification requests and documentation
- Verification status tracking (pending, verified, rejected)
- Administrative review records

Reporting and Analytics

- Usage statistics and platform metrics
- Opportunity popularity tracking
- Application success rates

2. Database Overview

The CampusLink platform uses a modern, cloud-hosted database backend to manage campus opportunities, streamline student-organization connections, and support real-time collaboration within the university community. The system ensures secure, reliable handling of student applications, organization profiles, opportunity postings, and verification processes. Key capabilities include:

● **Real-time Opportunity Management**: Supports live updates across organizations—such as new opportunity postings, application submissions, verification status changes, and profile updates—ensuring students always see the most current opportunities.

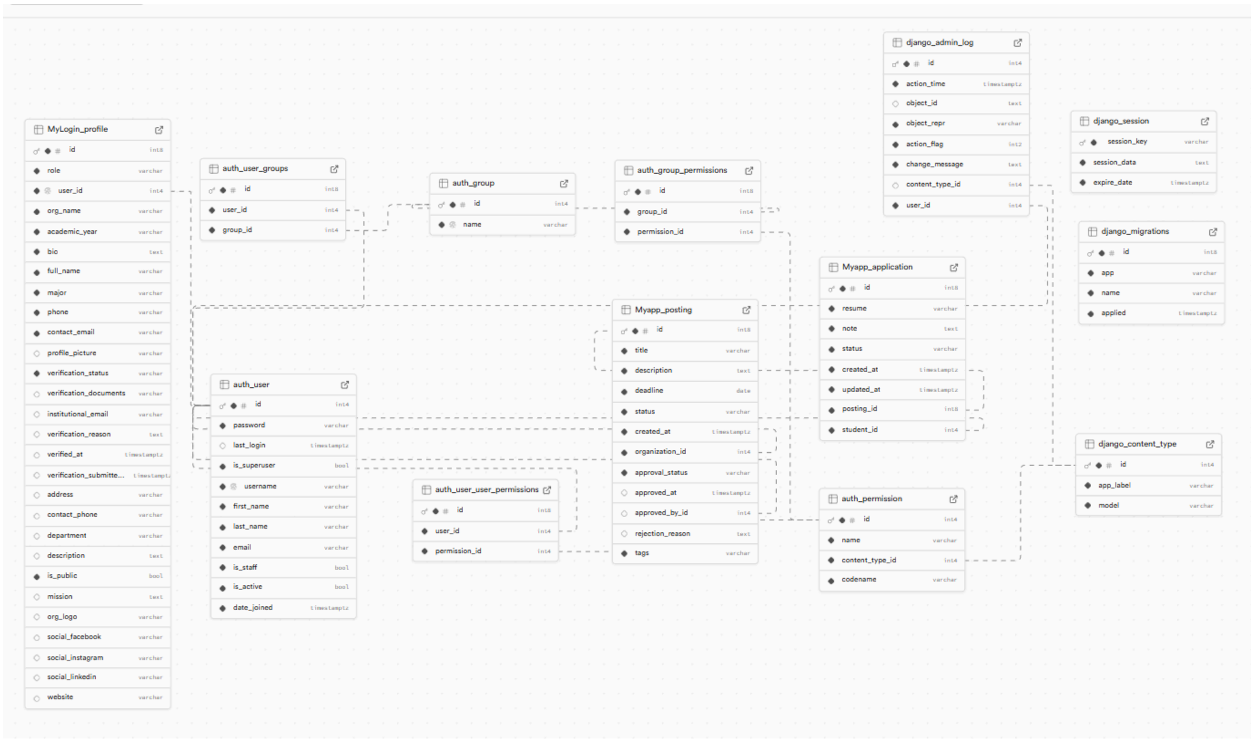
● **Data Integrity**: Maintains strong relational consistency using foreign key constraints linking students, organizations, opportunities, applications, and verification records. This ensures accurate, connected records throughout the platform.

● **Concurrent Access**: Allows admins, organization representatives, and students to use the system simultaneously with clearly defined role-based permissions controlling who can post opportunities, submit applications, verify organizations, or manage profiles.

● **Audit and Activity Tracking**: Captures essential logs such as opportunity creation, application submissions, verification requests, profile updates, and admin actions—providing transparency and accountability across all platform activities.

● **Cloud-Hosted Reliability**: Powered by Supabase with secure authentication, structured relational storage, and robust API support, ensuring fast response times, reliable uptime, and seamless access for all users. The system operates with session-based connection pooling to optimize resource usage while maintaining performance.

3. Entity-Relationship Diagram (ERD)



4. Database Schema

- MyLogin_profile
- Auth_user

- Auth_group
- Auth_user_groups
- Auth_permission
- Auth_group_permissions
- Auth_user_user_permissions
- Myapp_posting
- Myapp_application
- Django_admin_log
- Django_session
- Django_content_type
- django_migrations

5. Table Definitions

1. **auth_user Table**

Standard Django user authentication table storing basic user account information.

Columns:

- **id** (Integer, Primary Key): Unique identifier for each user
- **password** (Varchar): Hashed user password
- **last_login** (DateTime): Timestamp of last successful login
- **is_superuser** (Boolean): Superuser status
- **username** (Varchar): Unique username for login
- **first_name** (Varchar): User's first name
- **last_name** (Varchar): User's last name
- **email** (Varchar): User's email address
- **is_staff** (Boolean): Staff status for admin access
- **is_active** (Boolean): Account activation status
- **date_joined** (DateTime): Account creation timestamp

2. **MyLogin_profile Table**

Extended profile information for all user types in the CampusLink system.

Columns:

- **id** (Integer, Primary Key): Unique identifier for each profile
- **user_id** (Integer, Foreign Key): References auth_user(id)
- Organization-specific fields (org_name, department, description, contact information)
- Profile visibility settings (is_public)
- Verification status fields (verification_status, verification_submitted_at, verified_at, verification_reason)
- File paths for logos and profile pictures (org_logo, profile_picture)

3. **Myapp_posting Table**

Stores opportunity postings created by organizations.

Columns:

- id (Integer, Primary Key): Unique identifier for each posting
- Organization/owner reference
- Title and description fields
- Location and deadline information
- Team name (as required by specifications)
- Posting type classification
- Status flags
- Creation and modification timestamps

4. **Myapp_application Table**

Tracks student applications to opportunities.

Columns:

- id (Integer, Primary Key): Unique identifier for each application
- Student/user reference
- Posting reference
- Application status
- Submission timestamp
- Cover letter/resume fields
- Review/update timestamps

5. **django_session Table**

Django's session management table for maintaining user sessions.

Columns:

- **session_key** (Varchar, Primary Key): Unique session identifier
- **session_data** (Text): Serialized session data
- **expire_date** (DateTime): Session expiration timestamp

6. **django_migrations Table**

Tracks which database migrations have been applied.

Columns:

- **id** (Integer, Primary Key): Unique migration record identifier
- **app** (Varchar): Django app name
- **name** (Varchar): Migration name
- **applied** (DateTime): When migration was applied

7. **auth_permission Table**

Stores permission definitions for the Django authentication system.

Columns:

- **id** (Integer, Primary Key): Unique permission identifier
- **name** (Varchar): Human-readable name of the permission
- **content_type_id** (Integer, Foreign Key): References django_content_type(id)
- **codename** (Varchar): Machine-readable permission code name

8. **django_content_type Table**

Tracks content types for Django models and permissions.

Columns:

- **id** (Integer, Primary Key): Unique content type identifier
- **app_label** (Varchar): Django application label
- **model** (Varchar): Model name

9. **auth_group Table**

Stores user group definitions for permission grouping.

Columns:

- **id** (Integer, Primary Key): Unique group identifier
- **name** (Varchar): Group name

10. **auth_group_permissions Table**

Many-to-many relationship table between groups and permissions.

Columns:

- **id** (Integer, Primary Key): Unique relationship identifier
- **group_id** (Integer, Foreign Key): References auth_group(id)
- **permission_id** (Integer, Foreign Key): References auth_permission(id)

11. **auth_user_groups Table**

Many-to-many relationship table between users and groups.

Columns:

- **id** (Integer, Primary Key): Unique relationship identifier
- **user_id** (Integer, Foreign Key): References auth_user(id)
- **group_id** (Integer, Foreign Key): References auth_group(id)

12. **auth_user_user_permissions Table**

Direct user-to-permission assignments table.

Columns:

- **id** (Integer, Primary Key): Unique relationship identifier
- **user_id** (Integer, Foreign Key): References auth_user(id)
- **permission_id** (Integer, Foreign Key): References auth_permission(id)

13. **django_admin_log Table**

Logs admin actions in the Django admin interface.

Columns:

- **id** (Integer, Primary Key): Unique log entry identifier
- **action_time** (DateTime): When the action was performed
- **object_id** (Text): ID of the affected object
- **object_repr** (Varchar): String representation of the object
- **action_flag** (Integer): Type of action (add, change, delete)

- **change_message** (Text): Details of the change
- **content_type_id** (Integer, Foreign Key): References django_content_type(id)
- **user_id** (Integer, Foreign Key): References auth_user(id)

6. Relationships and Constraints

- **MyLogin_profile.id**
  Uniquely identifies each user profile record.
- **auth_user.id**
  Uniquely identifies each Django user account.
- **auth_group.id·**
  Uniquely identifies each user group.
- **auth_permission.id**
  Uniquely identifies each permission entry.
- **auth_user_groups *(composite key: user_id + group_id)***
  Uniquely identifies each user–group assignment.
- **auth_group_permissions *(composite key: group_id + permission_id)***
  Uniquely identifies each group–permission mapping.
- **auth_user_user_permissions *(composite key: user_id + permission_id)***
  Uniquely identifies each direct user–permission assignment.
- **Myapp_posting.id**
  Uniquely identifies each posting created by an organization.
- **Myapp_application.id**
  Uniquely identifies each student application to a posting.
- **django_admin_log.id**
  Uniquely identifies each admin audit log entry.
- **django_content_type.id**
  Uniquely identifies each Django model type.
- **django_migrations.id**
  Uniquely identifies each installed migration.
- **django_session.session_key**
  Uniquely identifies each user session.

FOREIGN KEY RELATIONSHIPS

MyLogin_profile Table

user_id → auth_user.id
Relationship: One-to-One / Many-to-One
Constraint: *ON DELETE CASCADE*
Meaning: When a Django user is deleted, their associated profile is removed.

auth_user_groups Table

user_id → auth_user.id
Relationship: Many-to-One
Constraint: *ON DELETE CASCADE*
Meaning: All group memberships of a user are deleted when the user is deleted.

group_id → auth_group.id
Relationship: Many-to-One
Constraint: *ON DELETE CASCADE*
Meaning: If a group is removed, all related user memberships are also removed.

auth_group_permissions Table

group_id → auth_group.id
Relationship: Many-to-One
Constraint: *ON DELETE CASCADE*

permission_id → auth_permission.id
Relationship: Many-to-One
Constraint: *ON DELETE CASCADE*

Meaning: When a group or permission is deleted, the mapping disappears automatically.

auth_user_user_permissions Table

user_id → auth_user.id
Relationship: Many-to-One
Constraint: *ON DELETE CASCADE*

permission_id → auth_permission.id
Relationship: Many-to-One
Constraint: *ON DELETE CASCADE*

Meaning: Direct user permissions are automatically removed when a user or permission is deleted.

Myapp_posting Table

organization_id → MyLogin_profile.id
Relationship: Many-to-One
Constraint: *ON DELETE CASCADE*
Meaning: If an organization profile is deleted, all their postings are also removed.

approved_by_id → auth_user.id
Relationship: Many-to-One
Constraint: *ON DELETE SET NULL*
Meaning: If an approving admin is deleted, the posting remains but approved_by becomes NULL.

Myapp_application Table

posting_id → Myapp_posting.id
Relationship: Many-to-One
Constraint: *ON DELETE CASCADE*
Meaning: Deleting a posting removes all student applications tied to it.

student_id → MyLogin_profile.id
Relationship: Many-to-One
Constraint: *ON DELETE CASCADE*

Meaning: If a student profile is deleted, their applications are removed as well.

django_admin_log Table

user_id → auth_user.id
Relationship: Many-to-One
Constraint: *ON DELETE SET NULL*
Meaning: If an admin user is deleted, log entries remain but the user reference becomes NULL.

content_type_id → django_content_type.id
Relationship: Many-to-One
Constraint: *ON DELETE SET NULL*
Meaning: Log entries persist even if the content type is removed.

django_permission Table

content_type_id → django_content_type.id
Relationship: Many-to-One
Constraint: *ON DELETE CASCADE*

Meaning: If a model type is deleted, all permissions tied to that model are also removed

7. Indexing Strategy

User Authentication Indexes:

- **Username Index**: Speeds up login queries and ensures fast duplicate username validation during registration.

- **Email Index**: Accelerates authentication lookups and prevents duplicate email registrations.

Profile Filtering Indexes:

- **Role Index**: Optimizes queries that filter users by role (Student, Organization, Admin) for dashboard displays.
- **Verification Status Index**: Improves performance when filtering organizations by verification status (pending, verified, rejected).

Opportunity Management Indexes:

- **Posting Type & Status Composite Index**: Accelerates filtering of opportunities by type (assistantship, volunteer, etc.) and active status. Deadline Index: Enhances sorting and retrieval of opportunities by deadline for time-sensitive displays.
- **Organization-Posting Relationship Index**: Speeds up queries linking postings to their originating organizations.

Application Processing Indexes:

- **Student-Opportunity Composite Index:** Optimizes application lookups, preventing duplicate submissions and speeding up verification of "Has this student already applied?"
- **Application Status Index**: Improves filtering of applications by status (submitted, reviewed, accepted, rejected).
  Submission Date Index: Enhances sorting and retrieval of recent applications.

Session Management Indexes:

- **Session Expiry Index:** Accelerates cleanup of expired sessions and validates active user sessions.

Composite Indexes on Common Query Patterns:

User Dashboard Queries:

- (user_id, role) - For loading role-specific dashboard content
- (verification_status, date_joined) - For admin verification dashboards

Opportunity Discovery:

- (posting_type, is_active, deadline) - For filtered opportunity listings
- (organization_id, is_active) - For organization-specific opportunity displays

Application Tracking:

- (student_id, submitted_at) - For student application history
- (posting_id, status) - For organization application review panels

A**dministrative Functions:**

- (content_type_id, object_id) - For admin log queries
- (user_id, action_time) - For user activity audits

8. Security Considerations

Authentication & Authorization

Password Security

- Password Hashing: Django's PBKDF2 + SHA256 algorithm for secure password storage
- Password Policy: Minimum 8 characters with mixed case and numbers
- Account Protection: Lockout after 5 failed attempts with progressive backoff
- Password Reset: Time-limited, single-use tokens sent to registered email

Role-Based Access Control (RBAC)

- System Admin: Full access to all system functions including user management and configuration
- Organization Representative: Manage organization profile, postings, and applicant reviews
- Student: Apply to opportunities, view public organization information
- Fine-Grained Permissions: CRUD operations restricted per role and enforced server-side

Account & Identity Verification

- University Affiliation: Email domain validation for student accounts
- Email Confirmation: Required before account activation
- Organization Verification: Document submission and admin approval process

Database Security

Access Control

- Principle of Least Privilege: Application database user has minimal required permissions
- Separation of Duties: Different database users for application and reporting functions
- Encrypted Connections: All database communications use SSL/TLS

Credential Management

- Environment Variables: Database credentials stored in .env files, not in source code
- Network Restrictions: VPC and IP whitelisting for administrative access
- Session Management: Automatic database session timeouts for admin functions

Data Protection & Privacy

Sensitive Information Handling

- Protected Fields: Personal information including names, emails, contact details, and passwords
- Encryption Standards: HTTPS-only communication with HSTS enforcement
- Password Storage: Salted hashes only, no plaintext storage

Privacy Compliance

- Data Minimization: Collection limited to essential fields only
- Retention Policy: Configurable data retention with automatic purging
- Legal Compliance: Adherence to Philippine Data Privacy Act RA 10173
- User Rights: Consent workflows, data export, and deletion capabilities

Audit & Logging

Activity Tracking

- Comprehensive Logs: User actions, login attempts, role changes, and data modifications
- Log Details: User ID, IP address, timestamp, and action specifics
- Tamper Protection: Write-once or append-only storage for audit logs
- Anomaly Detection: Alerts for suspicious activities including repeated failed logins

Backup & Recovery

Data Protection

- Encrypted Backups: Daily AES-256 encrypted database and file storage backups
- Retention Schedule: 7 daily, 4 weekly, 3 monthly snapshots
- Disaster Recovery: Cross-region backup copies for business continuity
- Access Control: Limited backup access with audit trails
- Recovery Testing: Quarterly restore drills to verify backup integrity

SQL Injection & Input Protection

Query Safety

- ORM Usage: Exclusive Django ORM usage with parameterized queries
- Input Validation: Strict server-side validation for all data inputs
- Raw Query Protection: Prepared statements for any necessary raw SQL
- File Sanitization: Validation and sanitization of uploaded file metadata

Session & API Security

Session Management

- Secure Cookies: HttpOnly, SameSite, and Secure flags enforced
- Timeout Policies: 30-minute inactivity timeout with absolute session limits
- CSRF Protection: Mandatory for all state-changing operations

API Security

- Token Authentication: JWT or opaque tokens with refresh mechanisms
- Rate Limiting: Per-IP and per-account throttling for abuse prevention
- CORS Controls: Restricted to approved university domains only

File Upload & Storage Security

Storage Protection

- Access Control: Signed URLs for secure file downloads
- Content Validation: File type and size limitations
- Filename Sanitization: Normalized and obfuscated uploaded filenames
- Metadata Separation: File metadata stored separately from content

Monitoring & Incident Response

System Oversight

- Real-time Monitoring: Performance metrics and anomaly detection
- Automated Alerts: Notifications for security events and unusual activities
- Regular Audits: Quarterly security assessments and vulnerability scanning
- Incident Response: Defined process for detection, containment, and recovery

Deployment & Operational Security

Infrastructure Security

- Infrastructure as Code: Peer-reviewed configuration changes
- CI/CD Security: Automated testing and security gates in deployment pipeline
- Secrets Management: Vault storage with regular credential rotation
- Administrative Access: Minimal access via jump hosts with MFA requirements

Third-Party & Integrations

Vendor Security

- Service Vetting: Compliance and data handling evaluation for all third-parties
- API Security: Least-privilege keys with usage monitoring
- Legal Agreements: Data processing agreements where required

Developer & Testing Practices

Secure Development

- Source Control: No secrets or credentials in repositories
- Test Data: Parameterized testing without production data
- Security Integration: Automated security checks in CI pipeline
- Training: Secure coding guidelines in developer onboarding

9. Team

Name	Role
Francis Kyle Mahinay	Lead Developer
John Joshua L. Meñez	Developer
Nicco Victor P. Maldo	Developer

10. Approval Sign-off

	Full Name	Signature	Date
Prepared By:	John Joshua L. Meñez	[illegible]	12/6/2025
Prepared By:	Francis Kyle G. Mahinay	[illegible]	12/6/2025
Prepared By:	Nicco Victor P. Maldo	[illegible]	12/6/2025
Reviewed By:	Jorge Martin Ogang	[illegible]	12/6/2025
Reviewed By:	Josh Anton Nuevas	[illegible]	12/6/2025
Reviewed By:	Florence Azriel R. Migallos	[illegible]	12/6/2025
Approved By:	Mr. Joemarie C. Amparo		12/6/2025
Approved By:	Mr. Frederick Revilleza		12/6/2025