Student Field Application System — Documentation

Kinondoni Municipal Council
Prepared by: Awesu N Mbaya
Date: August 2025

Title Page (copy to first page)

Student Field Application System
Kinondoni Municipal Council

Prepared by: Awesu N Mbaya
Date: August 2025

---

1. Executive Summary

This document describes the Student Field Application System designed for Kinondoni Municipal Council. The system streamlines student field placement applications by allowing students to apply online, upload documents, and select departments. The Head of Department (HoD), HR, and Admin manage and approve applications via dashboards. The outcome: reduced physical visits to the field station, faster approvals, centralized records, and better coordination.

---

2. Objectives

- Reduce the number of students physically visiting the field station/offices.
- Provide an online portal for student registration and document upload.
- Give department heads a way to set required student numbers.
- Enable HR to review, edit, approve, and issue field letters digitally.
- Provide admin full control and an audit trail.

---

3. Challenges (Current / Before System)

- Large numbers of students visiting the council to submit paper applications.
- Long queues and congestion at the field station.
- Manual data entry causing duplication and errors.
- Slow communication between departments, HR and students.

- Difficulty tracking which students were approved and for which department.

---

4. Problem Solving — How the System Reduces Student Movement

- **Online submission:** Students upload all documents remotely; no need to travel for submission.
- **Department quotas:** HoD sets required numbers so only selected students are asked to come (or none).
- **Email notifications & downloadable letters:** HR sends acceptance letters by email or the portal, reducing trips to collect physical letters.
- **Status tracking:** Students view application status online (pending / approved / rejected).
- **Scheduled pick-up / staggered visits:** If physical verification/pick-up is necessary, the system can produce time-slots to stagger student visits and avoid peaks.

---

5. System Architecture (Text)

Three-tier architecture:

1. **Presentation Layer:** Web UI (Django templates or React) — pages for Student, HoD, HR, Admin.
2. **Application Layer:** Django backend — handles authentication, business logic, validation, emailing.
3. **Data Layer:** Relational DB (PostgreSQL / MySQL) — stores users, profiles, documents, departments, applications, approvals.

Major modules:

- Authentication & Authorization
- Student Profile & Document Upload
- Department Management (HoD)
- HR Dashboard & Email Letter Generation
- Admin Management & Audit Logs

---

6. System Architecture Diagram (PlantUML)

You can render the following PlantUML to PNG/SVG using an online PlantUML editor or local PlantUML tool.

```
rust
```

```
@startuml
title Student Field Application System - Architecture

actor Student
actor HeadOfDept as HoD
actor HR
actor Admin

rectangle "Web App (Frontend)" {
  Student --> "Student UI"
  HoD --> "HoD UI"
  HR --> "HR UI"
  Admin --> "Admin UI"
}

rectangle "Django Backend (Application Layer)" {
  "Student UI" --> Backend
  "HoD UI" --> Backend
  "HR UI" --> Backend
  "Admin UI" --> Backend

  Backend --> "Auth Service"
  Backend --> "Application Service"
  Backend --> "Document Service"
  Backend --> "Notification/Email Service"
}

rectangle "Database (PostgreSQL/MySQL)" {
  Backend --> Database
}
Backend --> "File Storage (Media folder / S3)"

@enduml
```

---

7. Use Case Diagram (PlantUML)

Render this PlantUML for a use-case diagram:

```
pgsql
@startuml
left to right direction
actor Student
actor HoD as "Head of Department"
actor HR
actor Admin

Student --> (Register)
Student --> (Upload Documents)
Student --> (Submit Application)
Student --> (View Application Status)

(HoD) --> (Set Department Quota)
(HoD) --> (Review Applications)

(HR) --> (View Department Quotas)
(HR) --> (Review & Approve Applications)
(HR) --> (Send Approval Letter)

(Admin) --> (Manage Users)
(Admin) --> (Manage Departments)
```

```
(Admin) --> (View Reports)

@enduml
```

---

8. Database Design — Tables & Fields

Below are recommended tables with core fields. Adjust lengths/types to your DB.

8.1 `users` (custom user)

- `id` INT PK AUTO_INCREMENT
- `username` VARCHAR
- `email` VARCHAR (unique)
- `password_hash` VARCHAR
- `role` ENUM('student','hod','hr','admin')
- `first_name` VARCHAR
- `last_name` VARCHAR
- `is_active` BOOLEAN
- `created_at` TIMESTAMP
- `updated_at` TIMESTAMP

8.2 `student_profiles`

- `id` INT PK
- `user_id` FK -> users(id)
- `age` INT
- `nationality` VARCHAR
- `gender` VARCHAR
- `university` VARCHAR
- `program` VARCHAR
- `university_id` VARCHAR
- `profile_updated_at` TIMESTAMP

8.3 `departments`

- `id` INT PK
- `name` VARCHAR

- `hod_user_id` FK -> users(id) (user with role = hod)
- `slots_required` INT DEFAULT 0
- `created_at` TIMESTAMP

8.4 `documents`

- `id` INT PK
- `student_profile_id` FK -> student_profiles(id)
- `document_type` VARCHAR (e.g., 'CV','Transcript','Letter')
- `file_path` VARCHAR (where file is stored)
- `uploaded_at` TIMESTAMP

8.5 `applications`

- `id` INT PK
- `student_profile_id` FK -> student_profiles(id)
- `department_id` FK -> departments(id)
- `status` ENUM('pending','approved','rejected')
- `submitted_at` TIMESTAMP
- `hr_comments` TEXT NULL

8.6 `approvals`

- `id` INT PK
- `application_id` FK -> applications(id)
- `approved_by` FK -> users(id) (hod or hr)
- `role` ENUM('hod','hr')
- `status` ENUM('approved','rejected')
- `comments` TEXT
- `action_date` TIMESTAMP

8.7 `email_queue` (optional)

- `id` INT PK
- `to_email` VARCHAR
- `subject` VARCHAR
- `body` TEXT

- `status` ENUM('queued','sent','failed')
- `created_at` TIMESTAMP
- `sent_at` TIMESTAMP NULL

---

9. SQL `CREATE TABLE` (example: simplified MySQL syntax)

You can paste these into your SQL client (adjust types for PostgreSQL):

```
sql
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(150) NOT NULL,
  email VARCHAR(255) NOT NULL UNIQUE,
  password_hash VARCHAR(255) NOT NULL,
  role VARCHAR(20) NOT NULL,
  first_name VARCHAR(100),
  last_name VARCHAR(100),
  is_active BOOLEAN DEFAULT TRUE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);

CREATE TABLE student_profiles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  age INT,
  nationality VARCHAR(100),
  gender VARCHAR(20),
  university VARCHAR(200),
  program VARCHAR(200),
  university_id VARCHAR(100),
  profile_updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

CREATE TABLE departments (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(200),
  hod_user_id INT,
  slots_required INT DEFAULT 0,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (hod_user_id) REFERENCES users(id)
);

CREATE TABLE documents (
  id INT AUTO_INCREMENT PRIMARY KEY,
  student_profile_id INT,
  document_type VARCHAR(100),
  file_path VARCHAR(500),
  uploaded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (student_profile_id) REFERENCES student_profiles(id) ON DELETE
CASCADE
);

CREATE TABLE applications (
  id INT AUTO_INCREMENT PRIMARY KEY,
  student_profile_id INT,
  department_id INT,
```

```
  status VARCHAR(20) DEFAULT 'pending',
  submitted_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  hr_comments TEXT,
  FOREIGN KEY (student_profile_id) REFERENCES student_profiles(id) ON DELETE
CASCADE,
  FOREIGN KEY (department_id) REFERENCES departments(id)
);

CREATE TABLE approvals (
  id INT AUTO_INCREMENT PRIMARY KEY,
  application_id INT,
  approved_by INT,
  role VARCHAR(10),
  status VARCHAR(20),
  comments TEXT,
  action_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (application_id) REFERENCES applications(id),
  FOREIGN KEY (approved_by) REFERENCES users(id)
);
```

---

10. ERD (textual description + ASCII diagram)

Entities: Users, StudentProfiles, Departments, Applications, Documents, Approvals

Relations:

- `users` 1 — 1 `student_profiles` (only for role student)
- `departments` 1 — * `applications`
- `student_profiles` 1 — * `applications`
- `applications` 1 — * `approvals`
- `student_profiles` 1 — * `documents`

Simple ASCII ERD

```
less
[users] 1---1 [student_profiles]
  |
  |1
 [departments] 1---* [applications] *---1 [student_profiles]
                                  |
                                  *  [approvals]
[student_profiles] *---* [documents]
```

(Use a diagram tool like draw.io/diagrams.net or PlantUML to convert this into a nice visual.)

---

11. User Cases & Flows (Detailed)

11.1 Student: Register & Apply

1. Student registers with email & password.
2. Completes profile: name, age, nationality, gender, university, program, university ID.

3. Uploads required documents (CV, transcripts).
4. Chooses one or more departments to apply.
5. Submits application. Status = `pending`.
6. Receives email notifications when application is reviewed/approved/rejected.

Sequence (simpler): Register → Complete Profile → Upload Documents → Submit Application → Wait for HoD/HR decision → Receive letter.

11.2 HoD: Set Quota & Review

1. HoD logs in → sets `slots_required` for their department.
2. Views list of student applications for their department.
3. Shortlists or approves candidates (optional first-stage). If they approve, status progresses (could mark `hod_approved`).

11.3 HR: Review, Approve & Send Letters

1. HR gets aggregated quotas from HoDs.
2. HR reviews applications (HoD-approved or directly from queue).
3. HR approves final candidates, generates official letter (PDF) and sends by email or marks ready for pickup.
4. Application `status` updated to `approved` and entry added to `approvals`.

11.4 Admin

- Manage users, departments, view reports, reassign HoD, view logs.

---

12. Diagrams — How to get images

If you want images embedded in the PDF, do any of the following:

- Use PlantUML code above in an online PlantUML renderer (plantuml.com/plantuml) to get PNG/SVG.
- Use draw.io (diagrams.net) to draw architecture & ERD and export as PNG.
- Insert exported PNGs into your Word/LibreOffice document, then export PDF.

---

13. Installation & Running Instructions (short)

1. Unzip project.
2. Create Python venv: `python3 -m venv venv && source venv/bin/activate`

3. Install requirements: `pip install -r requirements.txt` (Django, etc.)
4. Apply migrations: `python manage.py migrate`
5. Create superuser: `python manage.py createsuperuser`
6. Run server: `python manage.py runserver`
7. Visit `http://127.0.0.1:8000/`

Notes: Configure `MEDIA_ROOT` and email settings in `settings.py` for production. Use PostgreSQL for production.

---

14. Recommendations for Reducing Student Movement at Field Station

- **Full online submission** with document verification via scanned documents and timestamped uploads.
- **Staggered collection windows**: If physical documents or badges must be collected, schedule date/time slots after approval.
- **Partial remote verification**: Use video or QR verification to confirm identity remotely.
- **Email & SMS notifications**: Inform only selected students to physically attend.
- **Reporting dashboard**: Show how many students per day/week to anticipate peaks and allocate staff.

---

15. Appendix — Sample email template (HR approval)

Subject: Kinondoni Municipal Council — Field Placement Approval

```
pgsql
Dear [Student Name],

Congratulations — your application for field placement in the [Department Name]
has been approved.

Please find your official placement letter attached. If physical pickup is
required, come to Kinondoni Municipal Council on [date] between [time-slot].

Regards,
HR
Kinondoni Municipal Council
```