**InterEd Admin Portal Pre Vibathon Preparation Roadmap**

**Overview**

This roadmap outlines a comprehensive plan to complete 70-80% of the foundational work for the InterEd Student Recruitment Admin Portal before the 48-hour Vibathon scheduled for May 6th, 2025. By following this structured approach, you'll maximize productivity during the Vibathon by having all critical architecture, design, and foundational elements ready in advance.

**Pre-Vibathon Preparation Timeline**

| **Timeframe** | **Focus** | **Completion Target** |
| --- | --- | --- |
| 4 Weeks Before Vibathon | Project Setup & Environment Configuration | 100% |
| 3 Weeks Before Vibathon | Architecture & Database Design | 90% |
| 2 Weeks Before Vibathon | Component Library & API Specification | 80% |
| 1 Week Before Vibathon | Authentication & Core Feature Implementation | 70% |
| Day Before Vibathon | Final Testing & Documentation | 100% |

**Detailed Work Breakdown**

**1. Project Setup & Environment Configuration (100% Completion)**

**1.1 Project Repository & Structure**

* Create GitHub repository with appropriate branch protection
* Set up project directory structure following MERN stack best practices
* Configure Git workflow (branch naming, PR templates, etc.)
* Create comprehensive README.md with project overview

**1.2 Development Environment**

* Set up Docker configuration for consistent development environment
* Configure CI/CD pipeline with GitHub Actions
* Create environment variable templates (.env.example)
* Set up VSCode configuration for team (extensions.json, settings.json)

**1.3 Tool Selection & Integration**

* Evaluate and select frontend tools (Cursor, Replit, Lovable)
* Set up Supabase project and initial configuration
* Configure Vercel for deployment
* Set up GitHub Copilot for development team

**1.4 Project Management**

* Create detailed project board in GitHub Projects
* Define key milestones and deliverables
* Set up communication channels for team
* Prepare documentation templates

**2. Architecture & Database Design (90% Completion)**

**2.1 System Architecture**

* Create detailed system architecture diagram
* Define component interaction patterns
* Document API structure and authentication flow
* Specify integration points with external systems
* Design scalability and performance considerations

**2.2 Database Schema**

* Design comprehensive database schema for Supabase
* Create entity-relationship diagrams
* Define table structures, relationships, and constraints
* Set up indexing strategy for performance optimization
* Design migration strategy for schema changes

**2.3 State Management Architecture**

* Define Redux store structure
* Create slice architecture for different modules
* Design state normalization strategy
* Document action types and reducer patterns
* Specify middleware requirements

**2.4 Security Planning**

* Design authentication and authorization flow
* Set up JWT configuration and refresh token strategy
* Define role-based access control (RBAC) structure
* Plan for secure data storage and transmission
* Design Row-Level Security (RLS) policies for Supabase

**3. Component Library & API Specification (80% Completion)**

**3.1 Design System**

* Extract design tokens from UI requirements
* Create Tailwind CSS configuration with custom theme
* Design component hierarchy and composition patterns
* Define typography, color, and spacing systems
* Create design documentation and style guide

**3.2 Core UI Components**

* Build reusable atomic components (buttons, inputs, etc.)
* Create layout components (containers, grid systems)
* Design data display components (tables, cards, etc.)
* Implement form components with validation
* Create navigation components (sidebar, breadcrumbs)

**3.3 API Specification**

* Create OpenAPI/Swagger documentation for all endpoints
* Define request/response schemas
* Document authentication requirements
* Specify error handling patterns
* Create API client service structure

**3.4 Feature-Specific Components**

* Design dashboard components and widgets
* Create student management interface components
* Design agent management components
* Build application processing interface components
* Implement settings and configuration components

**4. Authentication & Core Feature Implementation (70% Completion)**

**4.1 Authentication Implementation**

* Set up JWT authentication with Supabase
* Implement login and registration flows
* Create password reset functionality
* Build role-based access control system
* Implement session management and persistence

**4.2 Core API Integration**

* Set up API client with Axios or similar
* Implement base API service with interceptors
* Create authentication service for token management
* Build error handling and retry logic
* Set up request/response transformation

**4.3 State Management Implementation**

* Configure Redux store with Redux Toolkit
* Implement core slices (auth, users, etc.)
* Set up Redux middleware (thunk, logger)
* Create selector patterns for optimized rendering
* Build action creators for common operations

**4.4 Dashboard Implementation**

* Create dashboard layout with responsive design
* Implement summary statistics components
* Build chart and graph components for analytics
* Create recent activity feed components
* Implement quick action components

**5. Final Testing & Documentation (100% Completion)**

**5.1 Testing Setup**

* Configure Jest and React Testing Library
* Set up Cypress for end-to-end testing
* Create test utilities and mocks
* Implement test database seeding
* Design test coverage requirements

**5.2 Documentation**

* Create comprehensive API documentation
* Document component library with examples
* Provide detailed setup instructions
* Create user flow diagrams
* Document database schema and relationships

**5.3 Pre-Vibathon Verification**

* Verify all environments and configurations
* Test core functionality
* Validate database schema and migrations
* Ensure all documentation is up-to-date
* Prepare onboarding materials for Vibathon participants

**Claude-to-Tool Workflow**

To effectively transfer work created with Claude to specialized tools during the Vibathon, follow these guidelines:

**Transferring to Frontend Tools**

**Cursor/Replit**

1. Export components and code from Claude as text files
2. Import into your project structure
3. Run linting and formatting
4. Test the components in isolation
5. Integrate with the application

**Lovable**

1. Export UI component designs and specifications from Claude
2. Import the design tokens and styles
3. Recreate components using the visual interface
4. Export code and integrate with the project

**Transferring to Backend Tools**

**Supabase**

1. Use SQL scripts generated by Claude to create tables
2. Import database schema via Supabase CLI
3. Configure Row-Level Security policies
4. Set up authentication providers
5. Configure storage buckets

**Transferring to Deployment Tools**

**Vercel**

1. Configure project according to specifications from Claude
2. Set up environment variables
3. Configure build and deployment settings
4. Set up preview environments for branches
5. Configure domain and SSL settings

**Integrating with GitHub Copilot**

1. Use prompts from Claude as comments to guide Copilot
2. Structure code with clear comments for better suggestions
3. Use partial implementations to guide Copilot to complete
4. Combine Claude's architecture with Copilot's implementation

**Percentage of Work Achievable with Claude**

| **Component** | **Achievable with Claude** | **Transfer Effort** |
| --- | --- | --- |
| Architecture Documentation | 95-100% | Low |
| Database Schema Design | 90-95% | Low-Medium |
| API Specification | 90-95% | Low |
| UI Component Design | 80-85% | Medium |
| State Management Structure | 85-90% | Low-Medium |
| Authentication Flow | 70-75% | Medium |
| Core Application Logic | 70-75% | Medium |
| Testing Framework | 80-85% | Low |
| Documentation | 90-95% | Low |

**Key Deliverables Before Vibathon**

* Complete system architecture documentation
* Fully designed database schema with SQL scripts
* API specification with OpenAPI/Swagger documentation
* Core UI component library with Tailwind CSS
* Authentication and authorization implementation
* State management setup with Redux Toolkit
* Base application structure with routing
* Testing framework configuration
* Comprehensive documentation

**Conclusion**

By completing the tasks outlined in this roadmap, you'll ensure that approximately 70-80% of the foundational work for the InterEd Admin Portal is completed before the Vibathon. This preparation will allow the team to focus on integration, advanced features, and polishing during the 48-hour sprint, significantly increasing the likelihood of a successful delivery.

The pre-Vibathon work provides the strong foundation needed for rapid development during the event, with clear pathways for transferring Claude-generated work to specialized tools for refinement and implementation.

**InterEd Admin Portal Database Schema Design**

**Overview**

This document defines the comprehensive database schema for the InterEd Student Recruitment Admin Portal using Supabase (PostgreSQL). The schema is designed to support all features identified in the technical architecture documentation while ensuring scalability, performance, and security.

**Database Architecture**

The database design follows these principles:

* Normalized schema to minimize redundancy
* Strategic denormalization for performance where needed
* Appropriate indexing for query optimization
* Row-Level Security (RLS) for fine-grained access control
* Foreign key constraints to maintain data integrity
* Extensible design to accommodate future requirements

**Entity Relationship Diagram**

┌─────────────┐ ┌───────────────┐ ┌───────────────┐

│ profiles │ │ students │ │ applications │

├─────────────┤ ├───────────────┤ ├───────────────┤

│ id │ │ id │ │ id │

│ user\_id │───┐ │ profile\_id │────┐ │ student\_id │

│ role │ │ │ agent\_id │─┐ │ │ program\_id │

│ first\_name │ │ │ academic\_info │ │ │ │ status │

│ last\_name │ │ │ preferences │ │ │ │ documents │

│ email │ │ │ test\_scores │ │ └──│ submitted\_at │

│ phone │ │ │ status │ │ │ decision\_at │

│ created\_at │ │ │ created\_at │ │ │ created\_at │

│ updated\_at │ │ │ updated\_at │ │ │ updated\_at │

└─────────────┘ │ └───────────────┘ │ └───────────────┘

│ │ │

│ │ │

│ ┌────────────┐ │ │

└───│ users │ │ │

├────────────┤ │ │

│ id │ │ │

│ email │ │ │

│ password │ │ │

│ last\_login │ │ │

│ created\_at │ │ │

└────────────┘ │ │

│ │

┌────────────────┐ ┌─────────────┐ │ │ ┌─────────────────┐

│ universities │ │ agents │ │ │ │ programs │

├────────────────┤ ├─────────────┤ │ │ ├─────────────────┤

│ id │ │ id │ │ │ │ id │

│ name │ │ profile\_id │────┘ └────│ university\_id │

│ country │ │ agency\_name │ │ name │

│ ranking │ │ commission │ │ level │

│ programs\_count │────│ students │ │ duration │

│ created\_at │ │ status │ │ tuition\_fee │

│ updated\_at │ │ created\_at │ │ created\_at │

└────────────────┘ │ updated\_at │ │ updated\_at │

└─────────────┘ └─────────────────┘

│

│ ┌───────────────┐

│ │ commissions │

│ ├───────────────┤

└────│ id │

│ agent\_id │

│ application\_id│

│ amount │

│ status │

│ created\_at │

│ updated\_at │

└───────────────┘

**Table Definitions**

**Authentication & Users**

**auth.users (Supabase Auth)**

Supabase's built-in authentication table - extended with profiles table.

**profiles**

Stores the common profile information for all user types.

CREATE TABLE profiles (

id UUID PRIMARY KEY REFERENCES auth.users(id),

role VARCHAR(20) NOT NULL CHECK (role IN ('admin', 'staff', 'agent', 'student')),

first\_name VARCHAR(100) NOT NULL,

last\_name VARCHAR(100) NOT NULL,

email VARCHAR(255) NOT NULL UNIQUE,

phone VARCHAR(20),

avatar\_url VARCHAR(255),

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Enable Row-Level Security

ALTER TABLE profiles ENABLE ROW LEVEL SECURITY;

-- RLS Policies

CREATE POLICY "Users can view their own profile"

ON profiles FOR SELECT

USING (auth.uid() = id);

CREATE POLICY "Users can update their own profile"

ON profiles FOR UPDATE

USING (auth.uid() = id);

CREATE POLICY "Admins can view all profiles"

ON profiles FOR SELECT

USING (

EXISTS (

SELECT 1 FROM profiles WHERE id = auth.uid() AND role = 'admin'

)

);

CREATE POLICY "Staff can view all profiles"

ON profiles FOR SELECT

USING (

EXISTS (

SELECT 1 FROM profiles WHERE id = auth.uid() AND role IN ('admin', 'staff')

)

);

-- Triggers

CREATE TRIGGER update\_profiles\_updated\_at

BEFORE UPDATE ON profiles

FOR EACH ROW

EXECUTE FUNCTION update\_updated\_at();

**Student Management**

**students**

Stores detailed information about prospective and current students.

CREATE TABLE students (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

profile\_id UUID NOT NULL REFERENCES profiles(id),

agent\_id UUID REFERENCES agents(id),

academic\_info JSONB NOT NULL DEFAULT '{}'::JSONB,

preferences JSONB NOT NULL DEFAULT '{}'::JSONB,

test\_scores JSONB NOT NULL DEFAULT '{}'::JSONB,

status VARCHAR(20) NOT NULL DEFAULT 'active' CHECK (status IN ('active', 'inactive', 'graduated', 'withdrawn')),

metadata JSONB,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Indexes

CREATE INDEX idx\_students\_profile\_id ON students(profile\_id);

CREATE INDEX idx\_students\_agent\_id ON students(agent\_id);

CREATE INDEX idx\_students\_status ON students(status);

-- Enable Row-Level Security

ALTER TABLE students ENABLE ROW LEVEL SECURITY;

-- RLS Policies

CREATE POLICY "Students can view their own data"

ON students FOR SELECT

USING (

profile\_id = auth.uid()

);

CREATE POLICY "Agents can view their referred students"

ON students FOR SELECT

USING (

agent\_id = (SELECT id FROM agents WHERE profile\_id = auth.uid())

);

CREATE POLICY "Admins and staff can view all students"

ON students FOR ALL

USING (

EXISTS (

SELECT 1 FROM profiles WHERE id = auth.uid() AND role IN ('admin', 'staff')

)

);

-- Triggers

CREATE TRIGGER update\_students\_updated\_at

BEFORE UPDATE ON students

FOR EACH ROW

EXECUTE FUNCTION update\_updated\_at();

**academic\_history**

Stores educational background information for students.

CREATE TABLE academic\_history (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

student\_id UUID NOT NULL REFERENCES students(id) ON DELETE CASCADE,

institution VARCHAR(255) NOT NULL,

degree VARCHAR(255) NOT NULL,

field\_of\_study VARCHAR(255) NOT NULL,

start\_date DATE NOT NULL,

end\_date DATE,

grade VARCHAR(50),

description TEXT,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Indexes

CREATE INDEX idx\_academic\_history\_student\_id ON academic\_history(student\_id);

-- Enable Row-Level Security

ALTER TABLE academic\_history ENABLE ROW LEVEL SECURITY;

-- RLS Policies (similar pattern to students table)

**Agent Management**

**agents**

Stores information about recruitment agents and agencies.

CREATE TABLE agents (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

profile\_id UUID NOT NULL REFERENCES profiles(id),

agency\_name VARCHAR(255) NOT NULL,

agency\_type VARCHAR(50) NOT NULL CHECK (agency\_type IN ('individual', 'company', 'sub\_agent')),

parent\_agent\_id UUID REFERENCES agents(id),

commission\_tier VARCHAR(20) DEFAULT 'standard' CHECK (commission\_tier IN ('standard', 'silver', 'gold', 'platinum')),

business\_info JSONB NOT NULL DEFAULT '{}'::JSONB,

verification\_status VARCHAR(20) NOT NULL DEFAULT 'pending' CHECK (verification\_status IN ('pending', 'verified', 'rejected')),

status VARCHAR(20) NOT NULL DEFAULT 'active' CHECK (status IN ('active', 'inactive', 'suspended')),

metadata JSONB,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Indexes

CREATE INDEX idx\_agents\_profile\_id ON agents(profile\_id);

CREATE INDEX idx\_agents\_parent\_agent\_id ON agents(parent\_agent\_id);

CREATE INDEX idx\_agents\_status ON agents(status);

CREATE INDEX idx\_agents\_verification\_status ON agents(verification\_status);

-- Enable Row-Level Security

ALTER TABLE agents ENABLE ROW LEVEL SECURITY;

-- RLS Policies

CREATE POLICY "Agents can view their own data"

ON agents FOR SELECT

USING (

profile\_id = auth.uid()

);

CREATE POLICY "Agents can view their sub-agents"

ON agents FOR SELECT

USING (

parent\_agent\_id = (SELECT id FROM agents WHERE profile\_id = auth.uid())

);

CREATE POLICY "Admins and staff can view all agents"

ON agents FOR ALL

USING (

EXISTS (

SELECT 1 FROM profiles WHERE id = auth.uid() AND role IN ('admin', 'staff')

)

);

-- Triggers

CREATE TRIGGER update\_agents\_updated\_at

BEFORE UPDATE ON agents

FOR EACH ROW

EXECUTE FUNCTION update\_updated\_at();

**University and Program Management**

**universities**

Stores information about partner universities.

CREATE TABLE universities (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

name VARCHAR(255) NOT NULL,

short\_name VARCHAR(50),

country VARCHAR(100) NOT NULL,

city VARCHAR(100) NOT NULL,

ranking INT,

logo\_url VARCHAR(255),

website VARCHAR(255),

description TEXT,

programs\_count INT DEFAULT 0,

metadata JSONB,

status VARCHAR(20) NOT NULL DEFAULT 'active' CHECK (status IN ('active', 'inactive')),

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Indexes

CREATE INDEX idx\_universities\_country ON universities(country);

CREATE INDEX idx\_universities\_status ON universities(status);

-- Enable Row-Level Security

ALTER TABLE universities ENABLE ROW LEVEL SECURITY;

-- RLS Policies

CREATE POLICY "Universities are viewable by all authenticated users"

ON universities FOR SELECT

USING (auth.role() IS NOT NULL);

CREATE POLICY "Admins and staff can manage universities"

ON universities FOR ALL

USING (

EXISTS (

SELECT 1 FROM profiles WHERE id = auth.uid() AND role IN ('admin', 'staff')

)

);

-- Triggers

CREATE TRIGGER update\_universities\_updated\_at

BEFORE UPDATE ON universities

FOR EACH ROW

EXECUTE FUNCTION update\_updated\_at();

**programs**

Stores educational programs offered by universities.

CREATE TABLE programs (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

university\_id UUID NOT NULL REFERENCES universities(id) ON DELETE CASCADE,

name VARCHAR(255) NOT NULL,

level VARCHAR(50) NOT NULL CHECK (level IN ('bachelor', 'master', 'phd', 'diploma', 'certificate')),

duration VARCHAR(50) NOT NULL,

tuition\_fee DECIMAL(10, 2),

fee\_currency VARCHAR(3) DEFAULT 'USD',

description TEXT,

requirements JSONB NOT NULL DEFAULT '{}'::JSONB,

status VARCHAR(20) NOT NULL DEFAULT 'active' CHECK (status IN ('active', 'inactive')),

metadata JSONB,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Indexes

CREATE INDEX idx\_programs\_university\_id ON programs(university\_id);

CREATE INDEX idx\_programs\_level ON programs(level);

CREATE INDEX idx\_programs\_status ON programs(status);

-- Enable Row-Level Security

ALTER TABLE programs ENABLE ROW LEVEL SECURITY;

-- RLS Policies (similar to universities table)

-- Triggers

CREATE TRIGGER update\_programs\_updated\_at

BEFORE UPDATE ON programs

FOR EACH ROW

EXECUTE FUNCTION update\_updated\_at();

**Application Management**

**applications**

Stores student applications to university programs.

CREATE TABLE applications (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

student\_id UUID NOT NULL REFERENCES students(id),

program\_id UUID NOT NULL REFERENCES programs(id),

agent\_id UUID REFERENCES agents(id),

status VARCHAR(50) NOT NULL DEFAULT 'draft' CHECK (status IN (

'draft', 'submitted', 'under\_review', 'additional\_info\_required',

'conditional\_offer', 'unconditional\_offer', 'accepted', 'rejected', 'withdrawn'

)),

application\_fee DECIMAL(10, 2),

fee\_currency VARCHAR(3) DEFAULT 'USD',

fee\_paid BOOLEAN DEFAULT FALSE,

submitted\_at TIMESTAMP WITH TIME ZONE,

decision\_at TIMESTAMP WITH TIME ZONE,

intake\_date DATE,

notes TEXT,

documents JSONB DEFAULT '[]'::JSONB,

metadata JSONB,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Indexes

CREATE INDEX idx\_applications\_student\_id ON applications(student\_id);

CREATE INDEX idx\_applications\_program\_id ON applications(program\_id);

CREATE INDEX idx\_applications\_agent\_id ON applications(agent\_id);

CREATE INDEX idx\_applications\_status ON applications(status);

CREATE INDEX idx\_applications\_submitted\_at ON applications(submitted\_at);

-- Enable Row-Level Security

ALTER TABLE applications ENABLE ROW LEVEL SECURITY;

-- RLS Policies

CREATE POLICY "Students can view their own applications"

ON applications FOR SELECT

USING (

student\_id IN (SELECT id FROM students WHERE profile\_id = auth.uid())

);

CREATE POLICY "Agents can view applications from their students"

ON applications FOR SELECT

USING (

agent\_id = (SELECT id FROM agents WHERE profile\_id = auth.uid())

);

CREATE POLICY "Admins and staff can manage all applications"

ON applications FOR ALL

USING (

EXISTS (

SELECT 1 FROM profiles WHERE id = auth.uid() AND role IN ('admin', 'staff')

)

);

-- Triggers

CREATE TRIGGER update\_applications\_updated\_at

BEFORE UPDATE ON applications

FOR EACH ROW

EXECUTE FUNCTION update\_updated\_at();

**application\_status\_history**

Tracks the history of application status changes.

CREATE TABLE application\_status\_history (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

application\_id UUID NOT NULL REFERENCES applications(id) ON DELETE CASCADE,

previous\_status VARCHAR(50) NOT NULL,

new\_status VARCHAR(50) NOT NULL,

changed\_by UUID NOT NULL REFERENCES profiles(id),

notes TEXT,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Indexes

CREATE INDEX idx\_application\_status\_history\_application\_id ON application\_status\_history(application\_id);

CREATE INDEX idx\_application\_status\_history\_changed\_by ON application\_status\_history(changed\_by);

-- Enable Row-Level Security

ALTER TABLE application\_status\_history ENABLE ROW LEVEL SECURITY;

-- RLS Policies (similar pattern to applications table)

**Document Management**

**documents**

Stores metadata about uploaded documents. Actual files are stored in Supabase Storage.

CREATE TABLE documents (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

owner\_id UUID NOT NULL REFERENCES profiles(id),

owner\_type VARCHAR(20) NOT NULL CHECK (owner\_type IN ('student', 'agent', 'university', 'admin', 'staff')),

related\_id UUID,

related\_type VARCHAR(20) CHECK (related\_type IN ('application', 'student', 'agent', 'university', 'program')),

document\_type VARCHAR(50) NOT NULL,

name VARCHAR(255) NOT NULL,

file\_path VARCHAR(255) NOT NULL,

file\_size INT NOT NULL,

file\_type VARCHAR(100) NOT NULL,

verification\_status VARCHAR(20) DEFAULT 'pending' CHECK (verification\_status IN ('pending', 'verified', 'rejected')),

verified\_by UUID REFERENCES profiles(id),

verified\_at TIMESTAMP WITH TIME ZONE,

metadata JSONB,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Indexes

CREATE INDEX idx\_documents\_owner\_id ON documents(owner\_id);

CREATE INDEX idx\_documents\_related\_id ON documents(related\_id);

CREATE INDEX idx\_documents\_document\_type ON documents(document\_type);

CREATE INDEX idx\_documents\_verification\_status ON documents(verification\_status);

-- Enable Row-Level Security

ALTER TABLE documents ENABLE ROW LEVEL SECURITY;

-- RLS Policies

CREATE POLICY "Users can view their own documents"

ON documents FOR SELECT

USING (

owner\_id = auth.uid()

);

CREATE POLICY "Students can view documents related to their applications"

ON documents FOR SELECT

USING (

related\_type = 'application' AND

related\_id IN (

SELECT id FROM applications WHERE student\_id IN (

SELECT id FROM students WHERE profile\_id = auth.uid()

)

)

);

CREATE POLICY "Agents can view documents related to their students"

ON documents FOR SELECT

USING (

related\_type = 'student' AND

related\_id IN (

SELECT id FROM students WHERE agent\_id = (

SELECT id FROM agents WHERE profile\_id = auth.uid()

)

)

);

CREATE POLICY "Admins and staff can manage all documents"

ON documents FOR ALL

USING (

EXISTS (

SELECT 1 FROM profiles WHERE id = auth.uid() AND role IN ('admin', 'staff')

)

);

-- Triggers

CREATE TRIGGER update\_documents\_updated\_at

BEFORE UPDATE ON documents

FOR EACH ROW

EXECUTE FUNCTION update\_updated\_at();

**Financial Management**

**commissions**

Tracks agent commissions for successful applications.

CREATE TABLE commissions (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

agent\_id UUID NOT NULL REFERENCES agents(id),

application\_id UUID NOT NULL REFERENCES applications(id),

amount DECIMAL(10, 2) NOT NULL,

currency VARCHAR(3) DEFAULT 'USD',

rate DECIMAL(5, 2) NOT NULL,

status VARCHAR(20) NOT NULL DEFAULT 'pending' CHECK (status IN ('pending', 'approved', 'paid', 'rejected')),

payment\_date TIMESTAMP WITH TIME ZONE,

invoice\_id UUID,

notes TEXT,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Indexes

CREATE INDEX idx\_commissions\_agent\_id ON commissions(agent\_id);

CREATE INDEX idx\_commissions\_application\_id ON commissions(application\_id);

CREATE INDEX idx\_commissions\_status ON commissions(status);

-- Enable Row-Level Security

ALTER TABLE commissions ENABLE ROW LEVEL SECURITY;

-- RLS Policies

CREATE POLICY "Agents can view their own commissions"

ON commissions FOR SELECT

USING (

agent\_id = (SELECT id FROM agents WHERE profile\_id = auth.uid())

);

CREATE POLICY "Admins and staff can manage all commissions"

ON commissions FOR ALL

USING (

EXISTS (

SELECT 1 FROM profiles WHERE id = auth.uid() AND role IN ('admin', 'staff')

)

);

-- Triggers

CREATE TRIGGER update\_commissions\_updated\_at

BEFORE UPDATE ON commissions

FOR EACH ROW

EXECUTE FUNCTION update\_updated\_at();

**Communication & Notifications**

**messages**

Stores internal communication between users.

CREATE TABLE messages (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

sender\_id UUID NOT NULL REFERENCES profiles(id),

recipient\_id UUID NOT NULL REFERENCES profiles(id),

subject VARCHAR(255) NOT NULL,

content TEXT NOT NULL,

read BOOLEAN DEFAULT FALSE,

read\_at TIMESTAMP WITH TIME ZONE,

related\_id UUID,

related\_type VARCHAR(20) CHECK (related\_type IN ('application', 'student', 'agent', 'university', 'program')),

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Indexes

CREATE INDEX idx\_messages\_sender\_id ON messages(sender\_id);

CREATE INDEX idx\_messages\_recipient\_id ON messages(recipient\_id);

CREATE INDEX idx\_messages\_read ON messages(read);

CREATE INDEX idx\_messages\_related\_id ON messages(related\_id);

-- Enable Row-Level Security

ALTER TABLE messages ENABLE ROW LEVEL SECURITY;

-- RLS Policies

CREATE POLICY "Users can view messages they sent or received"

ON messages FOR SELECT

USING (

sender\_id = auth.uid() OR recipient\_id = auth.uid()

);

CREATE POLICY "Users can insert messages they send"

ON messages FOR INSERT

WITH CHECK (

sender\_id = auth.uid()

);

CREATE POLICY "Users can update messages they received"

ON messages FOR UPDATE

USING (

recipient\_id = auth.uid()

);

-- No triggers needed since messages don't get updated except for the read status

**notifications**

Stores system notifications for users.

CREATE TABLE notifications (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

user\_id UUID NOT NULL REFERENCES profiles(id),

title VARCHAR(255) NOT NULL,

content TEXT NOT NULL,

type VARCHAR(50) NOT NULL,

read BOOLEAN DEFAULT FALSE,

read\_at TIMESTAMP WITH TIME ZONE,

related\_id UUID,

related\_type VARCHAR(20) CHECK (related\_type IN ('application', 'student', 'agent', 'university', 'program', 'document')),

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Indexes

CREATE INDEX idx\_notifications\_user\_id ON notifications(user\_id);

CREATE INDEX idx\_notifications\_read ON notifications(read);

CREATE INDEX idx\_notifications\_related\_id ON notifications(related\_id);

-- Enable Row-Level Security

ALTER TABLE notifications ENABLE ROW LEVEL SECURITY;

-- RLS Policies

CREATE POLICY "Users can view their own notifications"

ON notifications FOR SELECT

USING (

user\_id = auth.uid()

);

CREATE POLICY "Users can update their own notifications"

ON notifications FOR UPDATE

USING (

user\_id = auth.uid()

);

CREATE POLICY "System and admins can insert notifications"

ON notifications FOR INSERT

WITH CHECK (

EXISTS (

SELECT 1 FROM profiles WHERE id = auth.uid() AND role IN ('admin', 'staff')

)

);

-- No triggers needed since notifications don't get updated except for the read status

**Analytics & Reporting**

**activity\_logs**

Tracks user activities for audit and analytics purposes.

CREATE TABLE activity\_logs (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

user\_id UUID NOT NULL REFERENCES profiles(id),

action VARCHAR(50) NOT NULL,

entity\_type VARCHAR(50) NOT NULL,

entity\_id UUID,

ip\_address VARCHAR(50),

user\_agent TEXT,

metadata JSONB,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Indexes

CREATE INDEX idx\_activity\_logs\_user\_id ON activity\_logs(user\_id);

CREATE INDEX idx\_activity\_logs\_action ON activity\_logs(action);

CREATE INDEX idx\_activity\_logs\_entity\_type\_entity\_id ON activity\_logs(entity\_type, entity\_id);

CREATE INDEX idx\_activity\_logs\_created\_at ON activity\_logs(created\_at);

-- Enable Row-Level Security

ALTER TABLE activity\_logs ENABLE ROW LEVEL SECURITY;

-- RLS Policies

CREATE POLICY "Users can view their own activity logs"

ON activity\_logs FOR SELECT

USING (

user\_id = auth.uid()

);

CREATE POLICY "Admins can view all activity logs"

ON activity\_logs FOR SELECT

USING (

EXISTS (

SELECT 1 FROM profiles WHERE id = auth.uid() AND role = 'admin'

)

);

CREATE POLICY "System can insert activity logs"

ON activity\_logs FOR INSERT

WITH CHECK (true);

-- No updates allowed on activity logs

**reports**

Stores saved report configurations.

CREATE TABLE reports (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

name VARCHAR(255) NOT NULL,

description TEXT,

creator\_id UUID NOT NULL REFERENCES profiles(id),

report\_type VARCHAR(50) NOT NULL,

parameters JSONB NOT NULL DEFAULT '{}'::JSONB,

schedule VARCHAR(50),

last\_run\_at TIMESTAMP WITH TIME ZONE,

recipients JSONB,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Indexes

CREATE INDEX idx\_reports\_creator\_id ON reports(creator\_id);

CREATE INDEX idx\_reports\_report\_type ON reports(report\_type);

-- Enable Row-Level Security

ALTER TABLE reports ENABLE ROW LEVEL SECURITY;

-- RLS Policies

CREATE POLICY "Users can view their own reports"

ON reports FOR SELECT

USING (

creator\_id = auth.uid()

);

CREATE POLICY "Users can manage their own reports"

ON reports FOR ALL

USING (

creator\_id = auth.uid()

);

CREATE POLICY "Admins can view all reports"

ON reports FOR SELECT

USING (

EXISTS (

SELECT 1 FROM profiles WHERE id = auth.uid() AND role = 'admin'

)

);

-- Triggers

CREATE TRIGGER update\_reports\_updated\_at

BEFORE UPDATE ON reports

FOR EACH ROW

EXECUTE FUNCTION update\_updated\_at();

**System Configuration**

**settings**

Stores system-wide and user-specific settings.

CREATE TABLE settings (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

key VARCHAR(100) NOT NULL,

value JSONB NOT NULL,

scope VARCHAR(50) NOT NULL CHECK (scope IN ('global', 'user', 'role')),

scope\_id UUID,

description TEXT,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

UNIQUE(key, scope, scope\_id)

);

-- Indexes

CREATE INDEX idx\_settings\_key\_scope\_scope\_id ON settings(key, scope, scope\_id);

-- Enable Row-Level Security

ALTER TABLE settings ENABLE ROW LEVEL SECURITY;

-- RLS Policies

CREATE POLICY "Users can view global settings"

ON settings FOR SELECT

USING (

scope = 'global'

);

CREATE POLICY "Users can view their own settings"

ON settings FOR SELECT

USING (

scope = 'user' AND scope\_id = auth.uid()

);

CREATE POLICY "Users can view their role settings"

ON settings FOR SELECT

USING (

scope = 'role' AND scope\_id IN (

SELECT role::uuid FROM profiles WHERE id = auth.uid()

)

);

CREATE POLICY "Users can update their own settings"

ON settings FOR UPDATE

USING (

scope = 'user' AND scope\_id = auth.uid()

);

CREATE POLICY "Admins can manage all settings"

ON settings FOR ALL

USING (

EXISTS (

SELECT 1 FROM profiles WHERE id = auth.uid() AND role = 'admin'

)

);

-- Triggers

CREATE TRIGGER update\_settings\_updated\_at

BEFORE UPDATE ON settings

FOR EACH ROW

EXECUTE FUNCTION update\_updated\_at();

**Common Database Functions**

-- Function to update updated\_at timestamp

CREATE OR REPLACE FUNCTION update\_updated\_at()

RETURNS TRIGGER AS $

BEGIN

NEW.updated\_at = NOW();

RETURN NEW;

END;

$ LANGUAGE plpgsql;

-- Function to check if user has admin role

CREATE OR REPLACE FUNCTION is\_admin()

RETURNS BOOLEAN AS $

BEGIN

RETURN EXISTS (

SELECT 1 FROM profiles

WHERE id = auth.uid() AND role = 'admin'

);

END;

$ LANGUAGE plpgsql SECURITY DEFINER;

-- Function to check if user has staff role

CREATE OR REPLACE FUNCTION is\_staff()

RETURNS BOOLEAN AS $

BEGIN

RETURN EXISTS (

SELECT 1 FROM profiles

WHERE id = auth.uid() AND role IN ('admin', 'staff')

);

END;

$ LANGUAGE plpgsql SECURITY DEFINER;

-- Function to get student ID from profile ID

CREATE OR REPLACE FUNCTION get\_student\_id(profile\_id UUID)

RETURNS UUID AS $

DECLARE

student\_id UUID;

BEGIN

SELECT id INTO student\_id FROM students WHERE profile\_id = $1;

RETURN student\_id;

END;

$ LANGUAGE plpgsql SECURITY DEFINER;

-- Function to get agent ID from profile ID

CREATE OR REPLACE FUNCTION get\_agent\_id(profile\_id UUID)

RETURNS UUID AS $

DECLARE

agent\_id UUID;

BEGIN

SELECT id INTO agent\_id FROM agents WHERE profile\_id = $1;

RETURN agent\_id;

END;

$ LANGUAGE plpgsql SECURITY DEFINER;

**Supabase Storage Configuration**

The InterEd Admin Portal will utilize Supabase Storage for file management. The following buckets should be configured:

**Storage Buckets**

**documents Bucket**

Stores all uploaded documents (transcripts, passports, certificates, etc.)

Security Rules:

-- RLS for documents bucket

CREATE POLICY "Users can view their own documents"

ON storage.objects FOR SELECT

USING (

bucket\_id = 'documents' AND

(

-- Document owner can view

(auth.uid())::text = (storage.foldername(name))[1] OR

-- Admins and staff can view all documents

EXISTS (

SELECT 1 FROM profiles

WHERE id = auth.uid() AND role IN ('admin', 'staff')

) OR

-- Agents can view their students' documents

EXISTS (

SELECT 1 FROM students

WHERE profile\_id::text = (storage.foldername(name))[1]

AND agent\_id = (

SELECT id FROM agents WHERE profile\_id = auth.uid()

)

)

)

);

CREATE POLICY "Users can upload their own documents"

ON storage.objects FOR INSERT

WITH CHECK (

bucket\_id = 'documents' AND

(auth.uid())::text = (storage.foldername(name))[1]

);

CREATE POLICY "Admins and staff can manage all documents"

ON storage.objects FOR ALL

USING (

bucket\_id = 'documents' AND

EXISTS (

SELECT 1 FROM profiles

WHERE id = auth.uid() AND role IN ('admin', 'staff')

)

);

**avatars Bucket**

Stores user profile pictures.

Security Rules:

-- RLS for avatars bucket

CREATE POLICY "Users can view all avatars"

ON storage.objects FOR SELECT

USING (

bucket\_id = 'avatars'

);

CREATE POLICY "Users can upload and manage their own avatar"

ON storage.objects FOR ALL

WITH CHECK (

bucket\_id = 'avatars' AND

(auth.uid())::text = (storage.foldername(name))[1]

);

**universities Bucket**

Stores university logos and images.

Security Rules:

-- RLS for universities bucket

CREATE POLICY "Anyone can view university images"

ON storage.objects FOR SELECT

USING (

bucket\_id = 'universities'

);

CREATE POLICY "Admins and staff can manage university images"

ON storage.objects FOR ALL

WITH CHECK (

bucket\_id = 'universities' AND

EXISTS (

SELECT 1 FROM profiles

WHERE id = auth.uid() AND role IN ('admin', 'staff')

)

);

**Database Migrations Strategy**

For database schema migrations, we'll use the Supabase CLI to manage migrations:

1. Initialize migrations:

supabase migration new initial\_schema

1. Apply migrations:

supabase db push

1. Versioning strategy:
   * Semantic versioning for schema changes
   * One migration file per feature/change
   * Clear naming convention: YYYYMMDD\_feature\_name.sql
   * Include both up and down migrations when possible

**Performance Considerations**

* Appropriate indexing on frequently queried columns
* Denormalization of certain data for read performance (e.g., student academic info as JSONB)
* Use of database functions for complex operations
* Pagination for large result sets
* Proper database connection pooling in application code

**Security Considerations**

* Row-Level Security (RLS) policies for all tables
* Function-based security checks
* Separation of authentication and authorization
* Encrypted sensitive data
* Audit logging for all critical operations
* Principle of least privilege for database access

**Conclusion**

This database schema design provides a comprehensive foundation for the InterEd Student Recruitment Admin Portal. The schema is designed to be secure, performant, and extensible, following best practices for PostgreSQL and Supabase.

The design incorporates:

* Authentication and user management
* Student and agent profiles
* University and program information
* Application processing
* Document management
* Financial tracking
* Messaging and notifications
* Analytics and reporting

By implementing this schema, you will have approximately 90% of the database requirements complete before the Vibathon, allowing you to focus on application logic and integrations during the event.

**InterEd Admin Portal Core UI Component Library**

**Overview**

This document outlines the core UI component library for the InterEd Student Recruitment Admin Portal. These components form the foundation of the user interface and can be implemented before the Vibathon to accelerate development during the event. The component library follows a systematic approach based on Atomic Design principles and is designed to be built with React, Tailwind CSS, and potentially shadcn/ui components.

**Design System Foundations**

**Design Tokens**

Before implementing components, establish these design tokens in your Tailwind configuration:

// tailwind.config.js

module.exports = {

theme: {

extend: {

colors: {

primary: {

50: '#F0F9FF',

100: '#E0F2FE',

200: '#BAE6FD',

300: '#7DD3FC',

400: '#38BDF8',

500: '#0EA5E9', // Main primary color

600: '#0284C7',

700: '#0369A1',

800: '#075985',

900: '#0C4A6E',

},

secondary: {

50: '#ECFDF5',

100: '#D1FAE5',

200: '#A7F3D0',

300: '#6EE7B7',

400: '#34D399',

500: '#10B981', // Main secondary color

600: '#059669',

700: '#047857',

800: '#065F46',

900: '#064E3B',

},

gray: {

50: '#F9FAFB',

100: '#F3F4F6',

200: '#E5E7EB',

300: '#D1D5DB',

400: '#9CA3AF',

500: '#6B7280',

600: '#4B5563',

700: '#374151',

800: '#1F2937',

900: '#111827',

},

success: '#22C55E',

warning: '#F59E0B',

error: '#EF4444',

info: '#3B82F6',

},

fontFamily: {

sans: ['Inter', 'sans-serif'],

display: ['Lexend', 'sans-serif'],

mono: ['JetBrains Mono', 'monospace'],

},

fontSize: {

xs: ['0.75rem', { lineHeight: '1rem' }],

sm: ['0.875rem', { lineHeight: '1.25rem' }],

base: ['1rem', { lineHeight: '1.5rem' }],

lg: ['1.125rem', { lineHeight: '1.75rem' }],

xl: ['1.25rem', { lineHeight: '1.75rem' }],

'2xl': ['1.5rem', { lineHeight: '2rem' }],

'3xl': ['1.875rem', { lineHeight: '2.25rem' }],

'4xl': ['2.25rem', { lineHeight: '2.5rem' }],

'5xl': ['3rem', { lineHeight: '1' }],

},

borderRadius: {

'sm': '0.125rem',

DEFAULT: '0.25rem',

'md': '0.375rem',

'lg': '0.5rem',

'xl': '0.75rem',

'2xl': '1rem',

'3xl': '1.5rem',

'full': '9999px',

},

boxShadow: {

sm: '0 1px 2px 0 rgba(0, 0, 0, 0.05)',

DEFAULT: '0 1px 3px 0 rgba(0, 0, 0, 0.1), 0 1px 2px 0 rgba(0, 0, 0, 0.06)',

md: '0 4px 6px -1px rgba(0, 0, 0, 0.1), 0 2px 4px -1px rgba(0, 0, 0, 0.06)',

lg: '0 10px 15px -3px rgba(0, 0, 0, 0.1), 0 4px 6px -2px rgba(0, 0, 0, 0.05)',

xl: '0 20px 25px -5px rgba(0, 0, 0, 0.1), 0 10px 10px -5px rgba(0, 0, 0, 0.04)',

'2xl': '0 25px 50px -12px rgba(0, 0, 0, 0.25)',

inner: 'inset 0 2px 4px 0 rgba(0, 0, 0, 0.06)',

none: 'none',

},

},

},

plugins: [

require('@tailwindcss/forms'),

require('@tailwindcss/typography'),

],

};

**Component Categorization**

The component library follows Atomic Design principles:

1. **Atoms**: Basic building blocks (buttons, inputs, icons)
2. **Molecules**: Combinations of atoms (form fields, search bars)
3. **Organisms**: Complex UI components (data tables, dashboards)
4. **Templates**: Page layouts and structures
5. **Pages**: Complete page implementations

**Core Components Implementation**

**Atoms**

**Button Component**

import React from 'react';

import { cva } from 'class-variance-authority';

import { cn } from '@/lib/utils';

const buttonVariants = cva(

"inline-flex items-center justify-center rounded-md text-sm font-medium transition-colors focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-offset-2 disabled:opacity-50 disabled:pointer-events-none",

{

variants: {

variant: {

default: "bg-primary-600 text-white hover:bg-primary-700 focus-visible:ring-primary-500",

secondary: "bg-secondary-600 text-white hover:bg-secondary-700 focus-visible:ring-secondary-500",

outline: "border border-gray-300 bg-transparent hover:bg-gray-50 focus-visible:ring-gray-500",

ghost: "hover:bg-gray-100 hover:text-gray-900 focus-visible:ring-gray-500",

destructive: "bg-error text-white hover:bg-red-700 focus-visible:ring-red-500",

},

size: {

default: "h-10 py-2 px-4",

sm: "h-8 px-3 rounded-md text-xs",

lg: "h-12 px-6 rounded-md text-base",

icon: "h-10 w-10",

},

},

defaultVariants: {

variant: "default",

size: "default",

},

}

);

const Button = React.forwardRef(

({ className, variant, size, children, ...props }, ref) => {

return (

<button

className={cn(buttonVariants({ variant, size, className }))}

ref={ref}

{...props}

>

{children}

</button>

);

}

);

Button.displayName = "Button";

export { Button, buttonVariants };

**Input Component**

import React from 'react';

import { cn } from '@/lib/utils';

const Input = React.forwardRef(

({ className, type, error, ...props }, ref) => {

return (

<div className="relative">

<input

type={type}

className={cn(

"flex h-10 w-full rounded-md border border-gray-300 bg-white px-3 py-2 text-sm placeholder:text-gray-400 focus:outline-none focus:ring-2 focus:ring-primary-500 focus:border-transparent disabled:cursor-not-allowed disabled:opacity-50",

error && "border-error focus:ring-error",

className

)}

ref={ref}

{...props}

/>

{error && (

<p className="mt-1 text-sm text-error">{error}</p>

)}

</div>

);

}

);

Input.displayName = "Input";

export { Input };

**Select Component**

import React from 'react';

import { cn } from '@/lib/utils';

const Select = React.forwardRef(

({ className, children, error, ...props }, ref) => {

return (

<div className="relative">

<select

className={cn(

"flex h-10 w-full rounded-md border border-gray-300 bg-white px-3 py-2 text-sm focus:outline-none focus:ring-2 focus:ring-primary-500 focus:border-transparent disabled:cursor-not-allowed disabled:opacity-50",

error && "border-error focus:ring-error",

className

)}

ref={ref}

{...props}

>

{children}

</select>

{error && (

<p className="mt-1 text-sm text-error">{error}</p>

)}

</div>

);

}

);

Select.displayName = "Select";

export { Select };

**Checkbox Component**

import React from 'react';

import { cn } from '@/lib/utils';

const Checkbox = React.forwardRef(

({ className, label, error, ...props }, ref) => {

return (

<div className="flex items-start">

<div className="flex items-center h-5">

<input

type="checkbox"

className={cn(

"h-4 w-4 rounded border-gray-300 text-primary-600 focus:ring-primary-500",

error && "border-error focus:ring-error",

className

)}

ref={ref}

{...props}

/>

</div>

{label && (

<div className="ml-3 text-sm">

<label className="font-medium text-gray-700">{label}</label>

{error && (

<p className="text-error">{error}</p>

)}

</div>

)}

</div>

);

}

);

Checkbox.displayName = "Checkbox";

export { Checkbox };

**Badge Component**

import React from 'react';

import { cva } from 'class-variance-authority';

import { cn } from '@/lib/utils';

const badgeVariants = cva(

"inline-flex items-center rounded-full px-2.5 py-0.5 text-xs font-medium",

{

variants: {

variant: {

default: "bg-primary-100 text-primary-800",

secondary: "bg-secondary-100 text-secondary-800",

success: "bg-green-100 text-green-800",

error: "bg-red-100 text-red-800",

warning: "bg-yellow-100 text-yellow-800",

info: "bg-blue-100 text-blue-800",

outline: "bg-transparent border border-gray-300 text-gray-700",

},

},

defaultVariants: {

variant: "default",

},

}

);

function Badge({ className, variant, ...props }) {

return (

<span className={cn(badgeVariants({ variant }), className)} {...props} />

);

}

export { Badge, badgeVariants };

**Avatar Component**

import React from 'react';

import { cn } from '@/lib/utils';

function getInitials(name) {

if (!name) return '';

return name

.split(' ')

.map(n => n[0])

.join('')

.toUpperCase();

}

const Avatar = React.forwardRef(

({ className, src, alt, name, size = 'md', ...props }, ref) => {

const [error, setError] = React.useState(false);

const sizeClasses = {

sm: 'h-8 w-8 text-xs',

md: 'h-10 w-10 text-sm',

lg: 'h-12 w-12 text-base',

xl: 'h-16 w-16 text-lg'

};

return (

<div

className={cn(

'relative inline-flex items-center justify-center overflow-hidden rounded-full bg-gray-200',

sizeClasses[size],

className

)}

ref={ref}

{...props}

>

{src && !error ? (

<img

src={src}

alt={alt || name || 'Avatar'}

className="h-full w-full object-cover"

onError={() => setError(true)}

/>

) : (

<span className="font-medium text-gray-600">

{getInitials(name) || '?'}

</span>

)}

</div>

);

}

);

Avatar.displayName = 'Avatar';

export { Avatar };

**Molecules**

**FormField Component**

import React from 'react';

import { cn } from '@/lib/utils';

const FormField = React.forwardRef(

({ className, label, htmlFor, error, required, children, helpText, ...props }, ref) => {

return (

<div className={cn("space-y-1", className)} ref={ref} {...props}>

{label && (

<label

htmlFor={htmlFor}

className="block text-sm font-medium text-gray-700"

>

{label}

{required && <span className="text-error ml-1">\*</span>}

</label>

)}

{children}

{helpText && !error && (

<p className="text-sm text-gray-500">{helpText}</p>

)}

{error && (

<p className="text-sm text-error">{error}</p>

)}

</div>

);

}

);

FormField.displayName = "FormField";

export { FormField };

**SearchInput Component**

import React from 'react';

import { SearchIcon } from 'lucide-react';

import { Input } from './Input';

import { cn } from '@/lib/utils';

const SearchInput = React.forwardRef(

({ className, onChange, value, placeholder = "Search...", ...props }, ref) => {

return (

<div className="relative">

<div className="absolute inset-y-0 left-0 flex items-center pl-3 pointer-events-none">

<SearchIcon className="h-4 w-4 text-gray-400" />

</div>

<Input

ref={ref}

type="search"

className={cn("pl-10", className)}

placeholder={placeholder}

value={value}

onChange={onChange}

{...props}

/>

</div>

);

}

);

SearchInput.displayName = "SearchInput";

export { SearchInput };

**Alert Component**

import React from 'react';

import { cva } from 'class-variance-authority';

import { AlertCircle, CheckCircle, Info, X, AlertTriangle } from 'lucide-react';

import { cn } from '@/lib/utils';

const alertVariants = cva(

"relative w-full p-4 rounded-md border",

{

variants: {

variant: {

default: "bg-gray-50 border-gray-200 text-gray-800",

info: "bg-blue-50 border-blue-200 text-blue-800",

success: "bg-green-50 border-green-200 text-green-800",

warning: "bg-yellow-50 border-yellow-200 text-yellow-800",

error: "bg-red-50 border-red-200 text-red-800",

},

},

defaultVariants: {

variant: "default",

},

}

);

const icons = {

default: Info,

info: Info,

success: CheckCircle,

warning: AlertTriangle,

error: AlertCircle,

};

const Alert = React.forwardRef(

({ className, variant, title, children, onClose, ...props }, ref) => {

const Icon = icons[variant] || icons.default;

return (

<div

ref={ref}

className={cn(alertVariants({ variant }), className)}

{...props}

>

<div className="flex gap-3">

<Icon className="h-5 w-5 flex-shrink-0" />

<div className="flex-1">

{title && <h5 className="font-medium mb-1">{title}</h5>}

<div className="text-sm">{children}</div>

</div>

{onClose && (

<button

onClick={onClose}

className="text-gray-500 hover:text-gray-700"

aria-label="Close"

>

<X className="h-4 w-4" />

</button>

)}

</div>

</div>

);

}

);

Alert.displayName = "Alert";

export { Alert, alertVariants };

**Card Component**

import React from 'react';

import { cn } from '@/lib/utils';

const Card = React.forwardRef(

({ className, ...props }, ref) => {

return (

<div

ref={ref}

className={cn(

"rounded-lg border border-gray-200 bg-white shadow-sm",

className

)}

{...props}

/>

);

}

);

Card.displayName = "Card";

const CardHeader = React.forwardRef(

({ className, ...props }, ref) => {

return (

<div

ref={ref}

className={cn("flex flex-col space-y-1.5 p-6", className)}

{...props}

/>

);

}

);

CardHeader.displayName = "CardHeader";

const CardTitle = React.forwardRef(

({ className, ...props }, ref) => {

return (

<h3

ref={ref}

className={cn("text-lg font-semibold leading-none tracking-tight", className)}

{...props}

/>

);

}

);

CardTitle.displayName = "CardTitle";

const CardDescription = React.forwardRef(

({ className

({ className, ...props }, ref) => { return ( <h3 ref={ref} className={cn("text-lg font-semibold leading-none tracking-tight", className)} {...props} /> ); } ); CardTitle.displayName = "CardTitle";

const CardDescription = React.forwardRef( ({ className, ...props }, ref) => { return ( <p ref={ref} className={cn("text-sm text-gray-500", className)} {...props} /> ); } ); CardDescription.displayName = "CardDescription";

const CardContent = React.forwardRef( ({ className, ...props }, ref) => { return ( <div ref={ref} className={cn("p-6 pt-0", className)} {...props} /> ); } ); CardContent.displayName = "CardContent";

const CardFooter = React.forwardRef( ({ className, ...props }, ref) => { return ( <div ref={ref} className={cn("flex items-center p-6 pt-0", className)} {...props} /> ); } ); CardFooter.displayName = "CardFooter";

export { Card, CardHeader, CardTitle, CardDescription, CardContent, CardFooter };

#### Tabs Component

```jsx

import React from 'react';

import { cn } from '@/lib/utils';

const Tabs = React.forwardRef(

({ className, defaultValue, value, onValueChange, children, ...props }, ref) => {

const [selectedValue, setSelectedValue] = React.useState(value || defaultValue);

const handleValueChange = (newValue) => {

setSelectedValue(newValue);

onValueChange?.(newValue);

};

const contextValue = React.useMemo(

() => ({

value: value !== undefined ? value : selectedValue,

onValueChange: handleValueChange,

}),

[value, selectedValue, handleValueChange]

);

return (

<TabsContext.Provider value={contextValue}>

<div ref={ref} className={cn("", className)} {...props}>

{children}

</div>

</TabsContext.Provider>

);

}

);

Tabs.displayName = "Tabs";

const TabsContext = React.createContext({});

const useTabsContext = () => {

const context = React.useContext(TabsContext);

if (!context) {

throw new Error("Tabs compound components must be used within a Tabs component");

}

return context;

};

const TabsList = React.forwardRef(

({ className, ...props }, ref) => {

return (

<div

ref={ref}

className={cn(

"inline-flex items-center justify-center rounded-md bg-gray-100 p-1",

className

)}

{...props}

/>

);

}

);

TabsList.displayName = "TabsList";

const TabsTrigger = React.forwardRef(

({ className, value, ...props }, ref) => {

const { value: selectedValue, onValueChange } = useTabsContext();

const isSelected = selectedValue === value;

return (

<button

ref={ref}

className={cn(

"inline-flex items-center justify-center whitespace-nowrap rounded-sm px-3 py-1.5 text-sm font-medium transition-all focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-offset-2 disabled:pointer-events-none disabled:opacity-50",

isSelected

? "bg-white text-primary-700 shadow-sm"

: "text-gray-600 hover:text-gray-900",

className

)}

onClick={() => onValueChange(value)}

{...props}

/>

);

}

);

TabsTrigger.displayName = "TabsTrigger";

const TabsContent = React.forwardRef(

({ className, value, ...props }, ref) => {

const { value: selectedValue } = useTabsContext();

const isSelected = selectedValue === value;

if (!isSelected) return null;

return (

<div

ref={ref}

className={cn("mt-2", className)}

{...props}

/>

);

}

);

TabsContent.displayName = "TabsContent";

export { Tabs, TabsList, TabsTrigger, TabsContent };

**Organisms**

**DataTable Component**

import React from 'react';

import {

useTable,

useSortBy,

usePagination,

useRowSelect,

useGlobalFilter,

} from 'react-table';

import { ChevronDown, ChevronUp, ChevronLeft, ChevronRight, ChevronsLeft, ChevronsRight } from 'lucide-react';

import { cn } from '@/lib/utils';

import { Button } from '../atoms/Button';

import { SearchInput } from '../molecules/SearchInput';

import { Select } from '../atoms/Select';

import { Checkbox } from '../atoms/Checkbox';

const DataTable = ({

columns,

data,

initialState = {},

onRowClick,

searchPlaceholder = "Search...",

noDataMessage = "No data found",

showSelection = false,

onSelectedRowsChange,

className,

}) => {

const {

getTableProps,

getTableBodyProps,

headerGroups,

prepareRow,

page,

canPreviousPage,

canNextPage,

pageOptions,

pageCount,

gotoPage,

nextPage,

previousPage,

setPageSize,

setGlobalFilter,

state: { pageIndex, pageSize, selectedRowIds, globalFilter },

} = useTable(

{

columns,

data,

initialState: { ...initialState },

},

useGlobalFilter,

useSortBy,

usePagination,

useRowSelect,

(hooks) => {

if (showSelection) {

hooks.visibleColumns.push((columns) => [

{

id: 'selection',

Header: ({ getToggleAllPageRowsSelectedProps }) => (

<Checkbox {...getToggleAllPageRowsSelectedProps()} />

),

Cell: ({ row }) => (

<Checkbox {...row.getToggleRowSelectedProps()} />

),

},

...columns,

]);

}

}

);

React.useEffect(() => {

onSelectedRowsChange?.(selectedRowIds);

}, [selectedRowIds, onSelectedRowsChange]);

return (

<div className="space-y-4">

<div className="flex justify-between">

<SearchInput

placeholder={searchPlaceholder}

value={globalFilter || ''}

onChange={(e) => setGlobalFilter(e.target.value)}

className="max-w-sm"

/>

<div className="flex items-center space-x-2">

<span className="text-sm text-gray-500">

Show

</span>

<Select

value={pageSize}

onChange={(e) => setPageSize(Number(e.target.value))}

className="h-8 w-16"

>

{[10, 20, 30, 40, 50].map((size) => (

<option key={size} value={size}>

{size}

</option>

))}

</Select>

</div>

</div>

<div className="rounded-md border">

<div className="overflow-x-auto">

<table

{...getTableProps()}

className={cn("w-full border-collapse", className)}

>

<thead className="bg-gray-50">

{headerGroups.map((headerGroup) => (

<tr {...headerGroup.getHeaderGroupProps()}>

{headerGroup.headers.map((column) => (

<th

{...column.getHeaderProps(column.getSortByToggleProps())}

className="group px-4 py-3 text-left text-sm font-medium text-gray-500 border-b border-gray-200"

>

<div className="flex items-center space-x-1">

<span>{column.render('Header')}</span>

<span className="flex flex-col">

{column.isSorted ? (

column.isSortedDesc ? (

<ChevronDown className="h-4 w-4" />

) : (

<ChevronUp className="h-4 w-4" />

)

) : (

<span className="opacity-0 group-hover:opacity-100 transition-opacity">

<ChevronUp className="h-4 w-4" />

</span>

)}

</span>

</div>

</th>

))}

</tr>

))}

</thead>

<tbody {...getTableBodyProps()} className="divide-y divide-gray-200">

{page.length > 0 ? (

page.map((row) => {

prepareRow(row);

return (

<tr

{...row.getRowProps()}

onClick={onRowClick ? () => onRowClick(row.original) : undefined}

className={cn(

"hover:bg-gray-50",

onRowClick && "cursor-pointer"

)}

>

{row.cells.map((cell) => (

<td

{...cell.getCellProps()}

className="px-4 py-3 text-sm text-gray-500"

>

{cell.render('Cell')}

</td>

))}

</tr>

);

})

) : (

<tr>

<td

colSpan={columns.length + (showSelection ? 1 : 0)}

className="px-4 py-8 text-center text-sm text-gray-500"

>

{noDataMessage}

</td>

</tr>

)}

</tbody>

</table>

</div>

</div>

<div className="flex items-center justify-between py-3">

<div className="flex items-center space-x-2">

<span className="text-sm text-gray-700">

Page <span className="font-medium">{pageIndex + 1}</span> of{" "}

<span className="font-medium">{pageOptions.length}</span>

</span>

</div>

<div className="flex items-center space-x-2">

<Button

variant="outline"

size="sm"

onClick={() => gotoPage(0)}

disabled={!canPreviousPage}

>

<ChevronsLeft className="h-4 w-4" />

</Button>

<Button

variant="outline"

size="sm"

onClick={() => previousPage()}

disabled={!canPreviousPage}

>

<ChevronLeft className="h-4 w-4" />

</Button>

<Button

variant="outline"

size="sm"

onClick={() => nextPage()}

disabled={!canNextPage}

>

<ChevronRight className="h-4 w-4" />

</Button>

<Button

variant="outline"

size="sm"

onClick={() => gotoPage(pageCount - 1)}

disabled={!canNextPage}

>

<ChevronsRight className="h-4 w-4" />

</Button>

</div>

</div>

</div>

);

};

export { DataTable };

**StatusCard Component**

import React from 'react';

import { cn } from '@/lib/utils';

import { Card, CardContent } from '../molecules/Card';

const StatusCard = ({

title,

value,

icon: Icon,

delta,

deltaType = 'neutral',

description,

className,

...props

}) => {

const deltaColor = {

positive: 'text-green-600',

negative: 'text-red-600',

neutral: 'text-gray-600',

};

const deltaIcon = {

positive: '↑',

negative: '↓',

neutral: '',

};

return (

<Card className={cn("overflow-hidden", className)} {...props}>

<CardContent className="p-6">

<div className="flex items-center justify-between">

<h3 className="text-base font-medium text-gray-900">{title}</h3>

{Icon && (

<div className="rounded-full bg-primary-50 p-2">

<Icon className="h-5 w-5 text-primary-600" />

</div>

)}

</div>

<div className="mt-4">

<p className="text-2xl font-semibold text-gray-900">{value}</p>

{delta && (

<p className="mt-2 flex items-center text-sm">

<span className={deltaColor[deltaType]}>

{deltaIcon[deltaType]} {delta}

</span>

{description && (

<span className="ml-2 text-gray-500">{description}</span>

)}

</p>

)}

</div>

</CardContent>

</Card>

);

};

export { StatusCard };

**SideNavigation Component**

import React from 'react';

import { cn } from '@/lib/utils';

import { ChevronDown } from 'lucide-react';

const SideNavigation = ({

className,

children,

...props

}) => {

return (

<nav

className={cn(

"w-64 flex-shrink-0 border-r border-gray-200 bg-white",

className

)}

{...props}

>

<div className="h-full overflow-y-auto py-4">

<ul className="space-y-1 px-2">

{children}

</ul>

</div>

</nav>

);

};

const NavItem = ({

href,

icon: Icon,

children,

active = false,

className,

...props

}) => {

return (

<li>

<a

href={href}

className={cn(

"flex items-center rounded-md px-3 py-2 text-sm font-medium",

active

? "bg-primary-50 text-primary-700"

: "text-gray-700 hover:bg-gray-100 hover:text-gray-900",

className

)}

{...props}

>

{Icon && <Icon className="mr-2 h-5 w-5 flex-shrink-0" />}

<span>{children}</span>

</a>

</li>

);

};

const NavGroup = ({

label,

icon: Icon,

children,

defaultOpen = false,

className,

...props

}) => {

const [open, setOpen] = React.useState(defaultOpen);

return (

<li className={className} {...props}>

<button

type="button"

className={cn(

"flex w-full items-center rounded-md px-3 py-2 text-sm font-medium text-gray-700 hover:bg-gray-100 hover:text-gray-900",

open && "bg-gray-100"

)}

onClick={() => setOpen(!open)}

>

{Icon && <Icon className="mr-2 h-5 w-5 flex-shrink-0" />}

<span className="flex-1 text-left">{label}</span>

<ChevronDown

className={cn(

"h-4 w-4 transition-transform",

open && "rotate-180"

)}

/>

</button>

{open && (

<ul className="mt-1 space-y-1 px-2">

{children}

</ul>

)}

</li>

);

};

SideNavigation.Item = NavItem;

SideNavigation.Group = NavGroup;

export { SideNavigation };

**Header Component**

import React from 'react';

import { cn } from '@/lib/utils';

import { BellIcon, MenuIcon, SearchIcon } from 'lucide-react';

import { Avatar } from '../atoms/Avatar';

const Header = ({

className,

onMenuClick,

profileImageUrl,

profileName,

onProfileClick,

onNotificationsClick,

notificationsCount = 0,

...props

}) => {

return (

<header

className={cn(

"flex h-16 items-center border-b border-gray-200 bg-white px-4 sm:px-6",

className

)}

{...props}

>

<button

type="button"

className="mr-4 md:hidden"

onClick={onMenuClick}

aria-label="Open menu"

>

<MenuIcon className="h-6 w-6 text-gray-500" />

</button>

<div className="flex flex-1 items-center justify-between">

<div className="flex flex-1">

<div className="relative max-w-md w-full">

<div className="pointer-events-none absolute inset-y-0 left-0 flex items-center pl-3">

<SearchIcon className="h-5 w-5 text-gray-400" />

</div>

<input

type="search"

placeholder="Search..."

className="block w-full rounded-md border border-gray-300 bg-white py-2 pl-10 pr-3 text-sm placeholder-gray-500 focus:border-primary-500 focus:outline-none focus:ring-1 focus:ring-primary-500"

/>

</div>

</div>

<div className="ml-4 flex items-center space-x-4">

<button

type="button"

className="relative rounded-full p-1 text-gray-400 hover:bg-gray-100 hover:text-gray-500 focus:outline-none focus:ring-2 focus:ring-primary-500"

onClick={onNotificationsClick}

aria-label="View notifications"

>

<BellIcon className="h-6 w-6" />

{notificationsCount > 0 && (

<span className="absolute top-0 right-0 block h-2 w-2 rounded-full bg-red-400 ring-2 ring-white" />

)}

</button>

<div className="relative">

<button

type="button"

className="flex items-center space-x-2 rounded-md text-sm focus:outline-none focus:ring-2 focus:ring-primary-500 focus:ring-offset-2"

onClick={onProfileClick}

aria-label="User menu"

>

<Avatar src={profileImageUrl} name={profileName} />

<span className="hidden md:flex md:items-center">

<span className="text-sm font-medium text-gray-700">{profileName}</span>

<ChevronDownIcon className="ml-1 h-4 w-4 text-gray-400" />

</span>

</button>

</div>

</div>

</div>

</header>

);

};

export { Header };

**Templates**

**DashboardLayout Component**

import React from 'react';

import { cn } from '@/lib/utils';

import { Header } from '../organisms/Header';

import { SideNavigation } from '../organisms/SideNavigation';

const DashboardLayout = ({

children,

navigation,

profileImageUrl,

profileName,

onProfileClick,

onNotificationsClick,

notificationsCount,

className,

...props

}) => {

const [sidebarOpen, setSidebarOpen] = React.useState(false);

return (

<div className="flex h-screen overflow-hidden" {...props}>

{/\* Sidebar for mobile \*/}

<div

className={cn(

"fixed inset-0 z-40 flex md:hidden",

sidebarOpen ? "visible" : "invisible"

)}

>

<div

className={cn(

"absolute inset-0 bg-gray-600 bg-opacity-75 transition-opacity",

sidebarOpen

? "ease-out duration-300 opacity-100"

: "ease-in duration-200 opacity-0"

)}

onClick={() => setSidebarOpen(false)}

/>

<div

className={cn(

"relative flex w-full max-w-xs flex-1 flex-col bg-white transition-transform",

sidebarOpen

? "ease-out duration-300 translate-x-0"

: "ease-in duration-200 -translate-x-full"

)}

>

<SideNavigation>

{navigation}

</SideNavigation>

</div>

</div>

{/\* Static sidebar for desktop \*/}

<div className="hidden md:flex md:flex-shrink-0">

<SideNavigation>

{navigation}

</SideNavigation>

</div>

{/\* Main content \*/}

<div className="flex flex-1 flex-col overflow-hidden">

<Header

onMenuClick={() => setSidebarOpen(true)}

profileImageUrl={profileImageUrl}

profileName={profileName}

onProfileClick={onProfileClick}

onNotificationsClick={onNotificationsClick}

notificationsCount={notificationsCount}

/>

<main className="flex-1 overflow-y-auto bg-gray-50 p-6">

<div className={cn("mx-auto max-w-7xl", className)}>

{children}

</div>

</main>

</div>

</div>

);

};

export { DashboardLayout };

**PageHeader Component**

import React from 'react';

import { cn } from '@/lib/utils';

import { Button } from '../atoms/Button';

const PageHeader = ({

title,

description,

actions,

breadcrumbs,

className,

...props

}) => {

return (

<div

className={cn("mb-8", className)}

{...props}

>

{breadcrumbs && (

<nav className="mb-4">

<ol className="flex items-center space-x-2 text-sm text-gray-500">

{breadcrumbs.map((breadcrumb, index) => (

<React.Fragment key={index}>

{index > 0 && <span>/</span>}

<li>

{breadcrumb.href ? (

<a

href={breadcrumb.href}

className="hover:text-gray-700"

>

{breadcrumb.label}

</a>

) : (

<span className="font-medium text-gray-900">

{breadcrumb.label}

</span>

)}

</li>

</React.Fragment>

))}

</ol>

</nav>

)}

<div className="flex flex-col-reverse sm:flex-row sm:items-center sm:justify-between">

<div>

<h1 className="text-2xl font-bold tracking-tight text-gray-900 sm:text-3xl">

{title}

</h1>

{description && (

<p className="mt-2 text-sm text-gray-500">

{description}

</p>

)}

</div>

{actions && (

<div className="mb-4 flex space-x-3 sm:mb-0">

{actions}

</div>

)}

</div>

</div>

);

};

export { PageHeader };

**ContentContainer Component**

import React from 'react';

import { cn } from '@/lib/utils';

import { Card } from '../molecules/Card';

const ContentContainer = ({

title,

description,

className,

children,

footer,

...props

}) => {

return (

<Card

className={cn("", className)}

{...props}

>

{(title || description) && (

<div className="border-b border-gray-200 px-6 py-4">

{title && (

<h2 className="text-lg font-medium text-gray-900">

{title}

</h2>

)}

{description && (

<p className="mt-1 text-sm text-gray-500">

{description}

</p>

)}

</div>

)}

<div className="px-6 py-5">

{children}

</div>

{footer && (

<div className="border-t border-gray-200 px-6 py-4">

{footer}

</div>

)}

</Card>

);

};

export { ContentContainer };

**Dashboard Components**

**Stats Components**

import React from 'react';

import { cn } from '@/lib/utils';

import { StatusCard } from '../organisms/StatusCard';

import {

Users,

FileText,

GraduationCap,

DollarSign,

PieChart

} from 'lucide-react';

const StatGrid = ({

stats,

className,

...props

}) => {

return (

<div

className={cn("grid grid-cols-1 gap-4 md:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4", className)}

{...props}

>

{stats.map((stat, index) => (

<StatusCard key={index} {...stat} />

))}

</div>

);

};

const DashboardStats = () => {

// This would normally fetch from an API

const stats = [

{

title: "Total Students",

value: "2,456",

icon: Users,

delta: "12%",

deltaType: "positive",

description: "from last month"

},

{

title: "Active Applications",

value: "843",

icon: FileText,

delta: "5%",

deltaType: "positive",

description: "from last month"

},

{

title: "Universities",

value: "48",

icon: GraduationCap,

delta: "2",

deltaType: "positive",

description: "new partnerships"

},

{

title: "Revenue",

value: "$125,430",

icon: DollarSign,

delta: "8%",

deltaType: "positive",

description: "from last month"

}

];

return <StatGrid stats={stats} />;

};

export { StatGrid, DashboardStats };

**RecentApplications Component**

import React from 'react';

import { DataTable } from '../organisms/DataTable';

import { Badge } from '../atoms/Badge';

import { Avatar } from '../atoms/Avatar';

import { formatDistanceToNow } from 'date-fns';

const statusColors = {

draft: 'default',

submitted: 'info',

under\_review: 'info',

additional\_info\_required: 'warning',

conditional\_offer: 'success',

unconditional\_offer: 'success',

accepted: 'success',

rejected: 'error',

withdrawn: 'default'

};

const formatDate = (date) => {

return formatDistanceToNow(new Date(date), { addSuffix: true });

};

const RecentApplications = ({ data = [] }) => {

const columns = React.useMemo(

() => [

{

Header: 'Student',

accessor: 'student',

Cell: ({ value }) => (

<div className="flex items-center">

<Avatar name={value.name} src={value.avatarUrl} size="sm" />

<div className="ml-4">

<div className="font-medium text-gray-900">{value.name}</div>

<div className="text-gray-500">{value.email}</div>

</div>

</div>

),

},

{

Header: 'Program',

accessor: 'program',

Cell: ({ value }) => (

<div>

<div className="font-medium">{value.name}</div>

<div className="text-gray-500">{value.university}</div>

</div>

),

},

{

Header: 'Status',

accessor: 'status',

Cell: ({ value }) => (

<Badge variant={statusColors[value] || 'default'}>

{value.replace(/\_/g, ' ')}

</Badge>

),

},

{

Header: 'Submitted',

accessor: 'submittedAt',

Cell: ({ value }) => formatDate(value),

},

{

Header: 'Agent',

accessor: 'agent',

Cell: ({ value }) => value?.name || 'Direct',

},

],

[]

);

return (

<DataTable

columns={columns}

data={data}

searchPlaceholder="Search applications..."

noDataMessage="No recent applications found"

onRowClick={(row) => {

// Navigate to application details

console.log('Navigate to application', row.id);

}}

/>

);

};

export { RecentApplications };

**Implementation Strategy**

To efficiently implement these components before the Vibathon, follow this strategy:

1. **Set up the design tokens first** - Configure your Tailwind CSS to include all the design tokens defined in this document.
2. **Implement in dependency order** - Start with atoms, move to molecules, then organisms, and finally templates. This way, each component can build on previously created ones.
3. **Use Storybook** - Implement Storybook to document and test components in isolation before integrating them into the full application.
4. **Test as you go** - Write basic unit tests for each component to ensure they function correctly before moving on.
5. **Create a component index** - Maintain an index file that exports all components for easy imports throughout the application.

**Integration with Frontend Tools**

**Transferring to Cursor/Replit**

1. Export the component code from this document as individual files
2. Create the appropriate directory structure in your project
3. Implement the required utilities (e.g., cn function for className merging)
4. Create an index file for easy imports

**Integration with Lovable**

1. Copy the design tokens and add them to your Lovable project
2. Create components using the visual interface, guided by the code in this document
3. Export the generated code and integrate it with your React components

**Using Components with GitHub Copilot**

When implementing pages with GitHub Copilot, provide clear comments that reference the components:

// Import components from the component library

import { DataTable } from '@/components/organisms/DataTable';

import { Button } from '@/components/atoms/Button';

import { PageHeader } from '@/components/templates/PageHeader';

import { ContentContainer } from '@/components/templates/ContentContainer';

// Create a student management page using the components

function StudentManagementPage() {

// Your implementation here

}

**Utility Functions**

To support the component library, implement these utility functions:

**cn Utility for Class Name Merging**

// lib/utils.js

import { clsx } from 'clsx';

import { twMerge } from 'tailwind-merge';

export function cn(...inputs) {

return twMerge(clsx(inputs));

}

**Date Formatting Utility**

// lib/dates.js

import { format, formatDistance, formatRelative } from 'date-fns';

export function formatDate(date, formatString = 'PP') {

return format(new Date(date), formatString);

}

export function formatDateRelative(date) {

return formatRelative(new Date(date), new Date());

}

export function formatDateDistance(date, options = {}) {

return formatDistance(new Date(date), new Date(), {

addSuffix: true,

...options,

});

}

**Component Library Structure**

The final structure of your component library should look like this:

components/

├── atoms/

│ ├── Avatar.jsx

│ ├── Badge.jsx

│ ├── Button.jsx

│ ├── Checkbox.jsx

│ ├── Input.jsx

│ ├── Select.jsx

│ └── index.js

├── molecules/

│ ├── Alert.jsx

│ ├── Card.jsx

│ ├── FormField.jsx

│ ├── SearchInput.jsx

│ ├── Tabs.jsx

│ └── index.js

├── organisms/

│ ├── DataTable.jsx

│ ├── Header.jsx

│ ├── SideNavigation.jsx

│ ├── StatusCard.jsx

│ └── index.js

├── templates/

│ ├── ContentContainer.jsx

│ ├── DashboardLayout.jsx

│ ├── PageHeader.jsx

│ └── index.js

├── dashboard/

│ ├── DashboardStats.jsx

│ ├── RecentApplications.jsx

│ └── index.js

└── index.js

**Pre-Vibathon Best Practices**

1. **Componentize everything** - Even small UI elements should be components to ensure consistency.
2. **Keep components pure** - Avoid adding business logic to components; use props and callbacks instead.
3. **Document with comments** - Add JSDoc comments to components to document their purpose and props.
4. **Use TypeScript for type safety** - If possible, implement the components with TypeScript for better IDE support and fewer runtime errors.
5. **Implement responsive design from the start** - All components should work on mobile, tablet, and desktop.
6. **Accessibility first** - Ensure all components meet WCAG 2.1 AA standards.
7. **Think about dark mode** - Design components with both light and dark modes in mind.

**Conclusion**

This core UI component library provides the essential building blocks for the InterEd Student Recruitment Admin Portal. By implementing these components before the Vibathon, you'll have a solid foundation to build upon during the event. The components are designed to be modular, reusable, and consistent, following modern React practices and Tailwind CSS styling.

During the Vibathon, these components can be easily integrated into the application, allowing the team to focus on implementing business logic and advanced features rather than spending time on basic UI elements.

Remember to set up your development environment with the required dependencies:

npm install clsx tailwind-merge lucide-react date-fns react-table class-variance-authority

With this component library in place, you'll be well-prepared to tackle the implementation of the InterEd Admin Portal during the Vibathon with a high degree of efficiency and quality.

**InterEd Admin Portal Core UI Component Library**

**Overview**

This document outlines the core UI component library for the InterEd Student Recruitment Admin Portal. These components form the foundation of the user interface and can be implemented before the Vibathon to accelerate development during the event. The component library follows a systematic approach based on Atomic Design principles and is designed to be built with React, Tailwind CSS, and potentially shadcn/ui components.

**Design System Foundations**

**Design Tokens**

Before implementing components, establish these design tokens in your Tailwind configuration:

// tailwind.config.js

module.exports = {

theme: {

extend: {

colors: {

primary: {

50: '#F0F9FF',

100: '#E0F2FE',

200: '#BAE6FD',

300: '#7DD3FC',

400: '#38BDF8',

500: '#0EA5E9', // Main primary color

600: '#0284C7',

700: '#0369A1',

800: '#075985',

900: '#0C4A6E',

},

secondary: {

50: '#ECFDF5',

100: '#D1FAE5',

200: '#A7F3D0',

300: '#6EE7B7',

400: '#34D399',

500: '#10B981', // Main secondary color

600: '#059669',

700: '#047857',

800: '#065F46',

900: '#064E3B',

},

gray: {

50: '#F9FAFB',

100: '#F3F4F6',

200: '#E5E7EB',

300: '#D1D5DB',

400: '#9CA3AF',

500: '#6B7280',

600: '#4B5563',

700: '#374151',

800: '#1F2937',

900: '#111827',

},

success: '#22C55E',

warning: '#F59E0B',

error: '#EF4444',

info: '#3B82F6',

},

fontFamily: {

sans: ['Inter', 'sans-serif'],

display: ['Lexend', 'sans-serif'],

mono: ['JetBrains Mono', 'monospace'],

},

fontSize: {

xs: ['0.75rem', { lineHeight: '1rem' }],

sm: ['0.875rem', { lineHeight: '1.25rem' }],

base: ['1rem', { lineHeight: '1.5rem' }],

lg: ['1.125rem', { lineHeight: '1.75rem' }],

xl: ['1.25rem', { lineHeight: '1.75rem' }],

'2xl': ['1.5rem', { lineHeight: '2rem' }],

'3xl': ['1.875rem', { lineHeight: '2.25rem' }],

'4xl': ['2.25rem', { lineHeight: '2.5rem' }],

'5xl': ['3rem', { lineHeight: '1' }],

},

borderRadius: {

'sm': '0.125rem',

DEFAULT: '0.25rem',

'md': '0.375rem',

'lg': '0.5rem',

'xl': '0.75rem',

'2xl': '1rem',

'3xl': '1.5rem',

'full': '9999px',

},

boxShadow: {

sm: '0 1px 2px 0 rgba(0, 0, 0, 0.05)',

DEFAULT: '0 1px 3px 0 rgba(0, 0, 0, 0.1), 0 1px 2px 0 rgba(0, 0, 0, 0.06)',

md: '0 4px 6px -1px rgba(0, 0, 0, 0.1), 0 2px 4px -1px rgba(0, 0, 0, 0.06)',

lg: '0 10px 15px -3px rgba(0, 0, 0, 0.1), 0 4px 6px -2px rgba(0, 0, 0, 0.05)',

xl: '0 20px 25px -5px rgba(0, 0, 0, 0.1), 0 10px 10px -5px rgba(0, 0, 0, 0.04)',

'2xl': '0 25px 50px -12px rgba(0, 0, 0, 0.25)',

inner: 'inset 0 2px 4px 0 rgba(0, 0, 0, 0.06)',

none: 'none',

},

},

},

plugins: [

require('@tailwindcss/forms'),

require('@tailwindcss/typography'),

],

};

**Component Categorization**

The component library follows Atomic Design principles:

1. **Atoms**: Basic building blocks (buttons, inputs, icons)
2. **Molecules**: Combinations of atoms (form fields, search bars)
3. **Organisms**: Complex UI components (data tables, dashboards)
4. **Templates**: Page layouts and structures
5. **Pages**: Complete page implementations

**Core Components Implementation**

**Atoms**

**Button Component**

import React from 'react';

import { cva } from 'class-variance-authority';

import { cn } from '@/lib/utils';

const buttonVariants = cva(

"inline-flex items-center justify-center rounded-md text-sm font-medium transition-colors focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-offset-2 disabled:opacity-50 disabled:pointer-events-none",

{

variants: {

variant: {

default: "bg-primary-600 text-white hover:bg-primary-700 focus-visible:ring-primary-500",

secondary: "bg-secondary-600 text-white hover:bg-secondary-700 focus-visible:ring-secondary-500",

outline: "border border-gray-300 bg-transparent hover:bg-gray-50 focus-visible:ring-gray-500",

ghost: "hover:bg-gray-100 hover:text-gray-900 focus-visible:ring-gray-500",

destructive: "bg-error text-white hover:bg-red-700 focus-visible:ring-red-500",

},

size: {

default: "h-10 py-2 px-4",

sm: "h-8 px-3 rounded-md text-xs",

lg: "h-12 px-6 rounded-md text-base",

icon: "h-10 w-10",

},

},

defaultVariants: {

variant: "default",

size: "default",

},

}

);

const Button = React.forwardRef(

({ className, variant, size, children, ...props }, ref) => {

return (

<button

className={cn(buttonVariants({ variant, size, className }))}

ref={ref}

{...props}

>

{children}

</button>

);

}

);

Button.displayName = "Button";

export { Button, buttonVariants };

**Input Component**

import React from 'react';

import { cn } from '@/lib/utils';

const Input = React.forwardRef(

({ className, type, error, ...props }, ref) => {

return (

<div className="relative">

<input

type={type}

className={cn(

"flex h-10 w-full rounded-md border border-gray-300 bg-white px-3 py-2 text-sm placeholder:text-gray-400 focus:outline-none focus:ring-2 focus:ring-primary-500 focus:border-transparent disabled:cursor-not-allowed disabled:opacity-50",

error && "border-error focus:ring-error",

className

)}

ref={ref}

{...props}

/>

{error && (

<p className="mt-1 text-sm text-error">{error}</p>

)}

</div>

);

}

);

Input.displayName = "Input";

export { Input };

**Select Component**

import React from 'react';

import { cn } from '@/lib/utils';

const Select = React.forwardRef(

({ className, children, error, ...props }, ref) => {

return (

<div className="relative">

<select

className={cn(

"flex h-10 w-full rounded-md border border-gray-300 bg-white px-3 py-2 text-sm focus:outline-none focus:ring-2 focus:ring-primary-500 focus:border-transparent disabled:cursor-not-allowed disabled:opacity-50",

error && "border-error focus:ring-error",

className

)}

ref={ref}

{...props}

>

{children}

</select>

{error && (

<p className="mt-1 text-sm text-error">{error}</p>

)}

</div>

);

}

);

Select.displayName = "Select";

export { Select };

**Checkbox Component**

import React from 'react';

import { cn } from '@/lib/utils';

const Checkbox = React.forwardRef(

({ className, label, error, ...props }, ref) => {

return (

<div className="flex items-start">

<div className="flex items-center h-5">

<input

type="checkbox"

className={cn(

"h-4 w-4 rounded border-gray-300 text-primary-600 focus:ring-primary-500",

error && "border-error focus:ring-error",

className

)}

ref={ref}

{...props}

/>

</div>

{label && (

<div className="ml-3 text-sm">

<label className="font-medium text-gray-700">{label}</label>

{error && (

<p className="text-error">{error}</p>

)}

</div>

)}

</div>

);

}

);

Checkbox.displayName = "Checkbox";

export { Checkbox };

**Badge Component**

import React from 'react';

import { cva } from 'class-variance-authority';

import { cn } from '@/lib/utils';

const badgeVariants = cva(

"inline-flex items-center rounded-full px-2.5 py-0.5 text-xs font-medium",

{

variants: {

variant: {

default: "bg-primary-100 text-primary-800",

secondary: "bg-secondary-100 text-secondary-800",

success: "bg-green-100 text-green-800",

error: "bg-red-100 text-red-800",

warning: "bg-yellow-100 text-yellow-800",

info: "bg-blue-100 text-blue-800",

outline: "bg-transparent border border-gray-300 text-gray-700",

},

},

defaultVariants: {

variant: "default",

},

}

);

function Badge({ className, variant, ...props }) {

return (

<span className={cn(badgeVariants({ variant }), className)} {...props} />

);

}

export { Badge, badgeVariants };

**Avatar Component**

import React from 'react';

import { cn } from '@/lib/utils';

function getInitials(name) {

if (!name) return '';

return name

.split(' ')

.map(n => n[0])

.join('')

.toUpperCase();

}

const Avatar = React.forwardRef(

({ className, src, alt, name, size = 'md', ...props }, ref) => {

const [error, setError] = React.useState(false);

const sizeClasses = {

sm: 'h-8 w-8 text-xs',

md: 'h-10 w-10 text-sm',

lg: 'h-12 w-12 text-base',

xl: 'h-16 w-16 text-lg'

};

return (

<div

className={cn(

'relative inline-flex items-center justify-center overflow-hidden rounded-full bg-gray-200',

sizeClasses[size],

className

)}

ref={ref}

{...props}

>

{src && !error ? (

<img

src={src}

alt={alt || name || 'Avatar'}

className="h-full w-full object-cover"

onError={() => setError(true)}

/>

) : (

<span className="font-medium text-gray-600">

{getInitials(name) || '?'}

</span>

)}

</div>

);

}

);

Avatar.displayName = 'Avatar';

export { Avatar };

**Molecules**

**FormField Component**

import React from 'react';

import { cn } from '@/lib/utils';

const FormField = React.forwardRef(

({ className, label, htmlFor, error, required, children, helpText, ...props }, ref) => {

return (

<div className={cn("space-y-1", className)} ref={ref} {...props}>

{label && (

<label

htmlFor={htmlFor}

className="block text-sm font-medium text-gray-700"

>

{label}

{required && <span className="text-error ml-1">\*</span>}

</label>

)}

{children}

{helpText && !error && (

<p className="text-sm text-gray-500">{helpText}</p>

)}

{error && (

<p className="text-sm text-error">{error}</p>

)}

</div>

);

}

);

FormField.displayName = "FormField";

export { FormField };

**SearchInput Component**

import React from 'react';

import { SearchIcon } from 'lucide-react';

import { Input } from './Input';

import { cn } from '@/lib/utils';

const SearchInput = React.forwardRef(

({ className, onChange, value, placeholder = "Search...", ...props }, ref) => {

return (

<div className="relative">

<div className="absolute inset-y-0 left-0 flex items-center pl-3 pointer-events-none">

<SearchIcon className="h-4 w-4 text-gray-400" />

</div>

<Input

ref={ref}

type="search"

className={cn("pl-10", className)}

placeholder={placeholder}

value={value}

onChange={onChange}

{...props}

/>

</div>

);

}

);

SearchInput.displayName = "SearchInput";

export { SearchInput };

**Alert Component**

import React from 'react';

import { cva } from 'class-variance-authority';

import { AlertCircle, CheckCircle, Info, X, AlertTriangle } from 'lucide-react';

import { cn } from '@/lib/utils';

const alertVariants = cva(

"relative w-full p-4 rounded-md border",

{

variants: {

variant: {

default: "bg-gray-50 border-gray-200 text-gray-800",

info: "bg-blue-50 border-blue-200 text-blue-800",

success: "bg-green-50 border-green-200 text-green-800",

warning: "bg-yellow-50 border-yellow-200 text-yellow-800",

error: "bg-red-50 border-red-200 text-red-800",

},

},

defaultVariants: {

variant: "default",

},

}

);

const icons = {

default: Info,

info: Info,

success: CheckCircle,

warning: AlertTriangle,

error: AlertCircle,

};

const Alert = React.forwardRef(

({ className, variant, title, children, onClose, ...props }, ref) => {

const Icon = icons[variant] || icons.default;

return (

<div

ref={ref}

className={cn(alertVariants({ variant }), className)}

{...props}

>

<div className="flex gap-3">

<Icon className="h-5 w-5 flex-shrink-0" />

<div className="flex-1">

{title && <h5 className="font-medium mb-1">{title}</h5>}

<div className="text-sm">{children}</div>

</div>

{onClose && (

<button

onClick={onClose}

className="text-gray-500 hover:text-gray-700"

aria-label="Close"

>

<X className="h-4 w-4" />

</button>

)}

</div>

</div>

);

}

);

Alert.displayName = "Alert";

export { Alert, alertVariants };

**Card Component**

import React from 'react';

import { cn } from '@/lib/utils';

const Card = React.forwardRef(

({ className, ...props }, ref) => {

return (

<div

ref={ref}

className={cn(

"rounded-lg border border-gray-200 bg-white shadow-sm",

className

)}

{...props}

/>

);

}

);

Card.displayName = "Card";

const CardHeader = React.forwardRef(

({ className, ...props }, ref) => {

return (

<div

ref={ref}

className={cn("flex flex-col space-y-1.5 p-6", className)}

{...props}

/>

);

}

);

CardHeader.displayName = "CardHeader";

const CardTitle = React.forwardRef(

({ className, ...props }, ref) => {

return (

<h3

ref={ref}

className={cn("text-lg font-semibold leading-none tracking-tight", className)}

{...props}

/>

);

}

);

CardTitle.displayName = "CardTitle";

const CardDescription = React.forwardRef(

({ className

**InterEd Admin Portal API Specification**

**Overview**

This document provides a comprehensive specification for the InterEd Student Recruitment Admin Portal API. It defines all endpoints, request/response formats, authentication mechanisms, and error handling patterns necessary for implementing the backend services for the admin portal.

The API is designed following RESTful principles with JWT authentication and is intended to be implemented using Express.js and integrated with Supabase.

**Base URL**

https://api.intered.com/v1

For development environments:

http://localhost:5000/api/v1

**Authentication**

**Authentication Mechanism**

The API uses JSON Web Tokens (JWT) for authentication:

1. All authenticated requests must include an Authorization header with a Bearer token
2. Tokens expire after 1 hour
3. Refresh tokens are used for obtaining new access tokens
4. Role-based permissions control access to resources

**Authentication Endpoints**

**POST /auth/login**

Authenticates a user and returns a JWT token.

**Request:**

{

"email": "admin@intered.com",

"password": "securepassword"

}

**Response:**

{

"success": true,

"data": {

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",

"refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",

"expiresIn": 3600,

"user": {

"id": "60d21b4667d0d8992e610c85",

"email": "admin@intered.com",

"firstName": "Admin",

"lastName": "User",

"role": "admin"

}

}

}

**POST /auth/refresh**

Refreshes an expired JWT token.

**Request:**

{

"refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."

}

**Response:**

{

"success": true,

"data": {

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",

"refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",

"expiresIn": 3600

}

}

**POST /auth/register**

Registers a new user (admin-only operation).

**Request:**

{

"email": "staff@intered.com",

"password": "securepassword",

"firstName": "Staff",

"lastName": "User",

"role": "staff"

}

**Response:**

{

"success": true,

"data": {

"user": {

"id": "60d21b4667d0d8992e610c85",

"email": "staff@intered.com",

"firstName": "Staff",

"lastName": "User",

"role": "staff"

}

}

}

**POST /auth/forgot-password**

Initiates password reset process.

**Request:**

{

"email": "admin@intered.com"

}

**Response:**

{

"success": true,

"message": "Password reset instructions sent to email"

}

**POST /auth/reset-password**

Resets password using reset token.

**Request:**

{

"token": "reset-token-from-email",

"newPassword": "newsecurepassword"

}

**Response:**

{

"success": true,

"message": "Password reset successfully"

}

**POST /auth/logout**

Invalidates the current JWT token.

**Response:**

{

"success": true,

"message": "Logged out successfully"

}

**Core API Endpoints**

**Users and Profile Management**

**GET /users**

Retrieves all users (admin-only).

**Query Parameters:**

* page (default: 1): Page number
* limit (default: 10): Items per page
* role: Filter by role (admin, staff, agent, student)
* search: Search by name or email
* sortBy: Field to sort by
* sortOrder: Sort order (asc, desc)

**Response:**

{

"success": true,

"data": {

"users": [

{

"id": "60d21b4667d0d8992e610c85",

"email": "admin@intered.com",

"firstName": "Admin",

"lastName": "User",

"role": "admin",

"lastLogin": "2025-05-01T10:30:00Z",

"status": "active",

"createdAt": "2025-01-01T10:00:00Z"

},

// Additional users...

],

"pagination": {

"page": 1,

"limit": 10,

"totalPages": 5,

"totalItems": 45

}

}

}

**GET /users/:id**

Retrieves a specific user.

**Response:**

{

"success": true,

"data": {

"user": {

"id": "60d21b4667d0d8992e610c85",

"email": "admin@intered.com",

"firstName": "Admin",

"lastName": "User",

"role": "admin",

"phone": "+1234567890",

"avatar": "https://storage.intered.com/avatars/admin.jpg",

"lastLogin": "2025-05-01T10:30:00Z",

"status": "active",

"createdAt": "2025-01-01T10:00:00Z",

"updatedAt": "2025-05-01T10:30:00Z"

}

}

}

**PUT /users/:id**

Updates a user.

**Request:**

{

"firstName": "Updated",

"lastName": "Name",

"phone": "+1234567890",

"status": "active"

}

**Response:**

{

"success": true,

"data": {

"user": {

"id": "60d21b4667d0d8992e610c85",

"firstName": "Updated",

"lastName": "Name",

"phone": "+1234567890",

"status": "active",

"updatedAt": "2025-05-01T10:35:00Z"

}

}

}

**DELETE /users/:id**

Deactivates a user (soft delete).

**Response:**

{

"success": true,

"message": "User deactivated successfully"

}

**POST /users/:id/avatar**

Uploads a user avatar.

**Request:** Form data with file field avatar

**Response:**

{

"success": true,

"data": {

"avatarUrl": "https://storage.intered.com/avatars/admin.jpg"

}

}

**Student Management**

**GET /students**

Retrieves all students.

**Query Parameters:**

* page (default: 1): Page number
* limit (default: 10): Items per page
* search: Search by name or email
* status: Filter by status (active, inactive, graduated, withdrawn)
* agentId: Filter by agent
* sortBy: Field to sort by
* sortOrder: Sort order (asc, desc)

**Response:**

{

"success": true,

"data": {

"students": [

{

"id": "60d21b4667d0d8992e610c86",

"profileId": "60d21b4667d0d8992e610c90",

"firstName": "John",

"lastName": "Doe",

"email": "john.doe@example.com",

"phone": "+1234567890",

"agentId": "60d21b4667d0d8992e610c95",

"agentName": "Global Education Agency",

"status": "active",

"applicationCount": 2,

"createdAt": "2025-01-15T10:00:00Z"

},

// Additional students...

],

"pagination": {

"page": 1,

"limit": 10,

"totalPages": 8,

"totalItems": 76

}

}

}

**GET /students/:id**

Retrieves a specific student with detailed information.

**Response:**

{

"success": true,

"data": {

"student": {

"id": "60d21b4667d0d8992e610c86",

"profileId": "60d21b4667d0d8992e610c90",

"firstName": "John",

"lastName": "Doe",

"email": "john.doe@example.com",

"phone": "+1234567890",

"dateOfBirth": "1998-05-15",

"nationality": "United States",

"address": {

"line1": "123 Main St",

"line2": "Apt 4B",

"city": "Boston",

"state": "MA",

"postalCode": "02115",

"country": "United States"

},

"academicInfo": {

"highestEducation": "High School",

"gpa": 3.8,

"institutions": [

{

"name": "Boston High School",

"degree": "High School Diploma",

"fieldOfStudy": "General",

"startDate": "2014-09-01",

"endDate": "2018-06-30",

"grade": "A"

}

]

},

"testScores": {

"ielts": {

"overall": 7.5,

"listening": 7.5,

"reading": 8.0,

"writing": 7.0,

"speaking": 7.5,

"testDate": "2024-11-15"

}

},

"preferences": {

"destinations": ["UK", "Canada", "Australia"],

"programLevels": ["Undergraduate"],

"fieldOfStudy": ["Computer Science", "Information Technology"],

"budget": {

"amount": 30000,

"currency": "USD"

}

},

"agent": {

"id": "60d21b4667d0d8992e610c95",

"name": "Global Education Agency",

"counselorName": "Jane Smith",

"email": "jane.smith@globaledu.com"

},

"status": "active",

"createdAt": "2025-01-15T10:00:00Z",

"updatedAt": "2025-04-20T14:30:00Z"

}

}

}

**POST /students**

Creates a new student.

**Request:**

{

"firstName": "Jane",

"lastName": "Smith",

"email": "jane.smith@example.com",

"phone": "+1987654321",

"dateOfBirth": "1999-03-20",

"nationality": "United Kingdom",

"address": {

"line1": "456 High St",

"city": "London",

"postalCode": "E1 6AN",

"country": "United Kingdom"

},

"academicInfo": {

"highestEducation": "High School",

"gpa": 4.0,

"institutions": [

{

"name": "London Academy",

"degree": "A-Levels",

"fieldOfStudy": "Sciences",

"startDate": "2016-09-01",

"endDate": "2018-06-30",

"grade": "A\*"

}

]

},

"agentId": "60d21b4667d0d8992e610c95"

}

**Response:**

{

"success": true,

"data": {

"student": {

"id": "60d21b4667d0d8992e610c87",

"email": "jane.smith@example.com",

"firstName": "Jane",

"lastName": "Smith",

"createdAt": "2025-05-01T11:00:00Z"

}

}

}

**PUT /students/:id**

Updates a student.

**Request:**

{

"phone": "+1987654322",

"address": {

"line1": "789 New Ave",

"city": "London",

"postalCode": "E2 8LP",

"country": "United Kingdom"

},

"preferences": {

"destinations": ["UK", "Canada"],

"programLevels": ["Undergraduate"],

"fieldOfStudy": ["Computer Science"]

}

}

**Response:**

{

"success": true,

"data": {

"student": {

"id": "60d21b4667d0d8992e610c87",

"updatedAt": "2025-05-01T11:30:00Z"

}

}

}

**DELETE /students/:id**

Deactivates a student (soft delete).

**Response:**

{

"success": true,

"message": "Student deactivated successfully"

}

**Agent Management**

**GET /agents**

Retrieves all agents.

**Query Parameters:**

* page (default: 1): Page number
* limit (default: 10): Items per page
* search: Search by name, agency name, or email
* status: Filter by status (active, inactive, suspended)
* verificationStatus: Filter by verification status (pending, verified, rejected)
* sortBy: Field to sort by
* sortOrder: Sort order (asc, desc)

**Response:**

{

"success": true,

"data": {

"agents": [

{

"id": "60d21b4667d0d8992e610c95",

"profileId": "60d21b4667d0d8992e610c96",

"firstName": "Jane",

"lastName": "Smith",

"email": "jane.smith@globaledu.com",

"phone": "+1234567890",

"agencyName": "Global Education Agency",

"agencyType": "company",

"verificationStatus": "verified",

"status": "active",

"studentCount": 15,

"applicationCount": 28,

"createdAt": "2024-12-01T10:00:00Z"

},

// Additional agents...

],

"pagination": {

"page": 1,

"limit": 10,

"totalPages": 3,

"totalItems": 25

}

}

}

**GET /agents/:id**

Retrieves a specific agent with detailed information.

**Response:**

{

"success": true,

"data": {

"agent": {

"id": "60d21b4667d0d8992e610c95",

"profileId": "60d21b4667d0d8992e610c96",

"firstName": "Jane",

"lastName": "Smith",

"email": "jane.smith@globaledu.com",

"phone": "+1234567890",

"agencyName": "Global Education Agency",

"agencyType": "company",

"businessInfo": {

"registrationNumber": "GEA12345",

"foundingYear": 2015,

"website": "https://www.globaledu.com",

"address": {

"line1": "100 Education St",

"city": "New York",

"state": "NY",

"postalCode": "10001",

"country": "United States"

}

},

"specializations": {

"countries": ["UK", "USA", "Canada", "Australia"],

"programLevels": ["Undergraduate", "Postgraduate"],

"disciplines": ["Business", "Engineering", "Computer Science"]

},

"verificationStatus": "verified",

"verificationDate": "2025-01-10T09:30:00Z",

"verifiedBy": "60d21b4667d0d8992e610c85",

"status": "active",

"commissionTier": "gold",

"parentAgentId": null,

"subAgents": [

{

"id": "60d21b4667d0d8992e610c97",

"name": "John Johnson",

"email": "john.johnson@globaledu.com",

"studentCount": 8

}

],

"performance": {

"studentCount": 15,

"applicationCount": 28,

"conversionRate": 85,

"commissionEarned": {

"amount": 25000,

"currency": "USD"

}

},

"agreementDetails": {

"startDate": "2025-01-01",

"endDate": "2025-12-31",

"documentId": "60d21b4667d0d8992e610c98"

},

"createdAt": "2024-12-01T10:00:00Z",

"updatedAt": "2025-04-15T15:30:00Z"

}

}

}

**POST /agents**

Creates a new agent.

**Request:**

{

"firstName": "Robert",

"lastName": "Johnson",

"email": "robert.johnson@studyabroad.com",

"phone": "+1122334455",

"agencyName": "Study Abroad Experts",

"agencyType": "company",

"businessInfo": {

"registrationNumber": "SAE54321",

"foundingYear": 2018,

"website": "https://www.studyabroad.com",

"address": {

"line1": "200 Education Blvd",

"city": "Chicago",

"state": "IL",

"postalCode": "60601",

"country": "United States"

}

},

"specializations": {

"countries": ["USA", "Canada"],

"programLevels": ["Undergraduate", "Postgraduate"],

"disciplines": ["Business", "Arts"]

}

}

**Response:**

{

"success": true,

"data": {

"agent": {

"id": "60d21b4667d0d8992e610c99",

"email": "robert.johnson@studyabroad.com",

"firstName": "Robert",

"lastName": "Johnson",

"agencyName": "Study Abroad Experts",

"verificationStatus": "pending",

"createdAt": "2025-05-01T12:30:00Z"

}

}

}

**PUT /agents/:id**

Updates an agent.

**Request:**

{

"phone": "+1122334466",

"businessInfo": {

"website": "https://www.studyabroadexperts.com",

"address": {

"line1": "300 Education Ctr",

"city": "Chicago",

"state": "IL",

"postalCode": "60602",

"country": "United States"

}

},

"specializations": {

"countries": ["USA", "Canada", "UK"],

"programLevels": ["Undergraduate", "Postgraduate", "Diploma"],

"disciplines": ["Business", "Arts", "Technology"]

}

}

**Response:**

{

"success": true,

"data": {

"agent": {

"id": "60d21b4667d0d8992e610c99",

"updatedAt": "2025-05-01T13:00:00Z"

}

}

}

**PUT /agents/:id/verify**

Verifies an agent (admin/staff only).

**Request:**

{

"verificationStatus": "verified",

"commissionTier": "silver",

"notes": "Documents verified successfully"

}

**Response:**

{

"success": true,

"data": {

"agent": {

"id": "60d21b4667d0d8992e610c99",

"verificationStatus": "verified",

"verificationDate": "2025-05-01T13:30:00Z",

"verifiedBy": "60d21b4667d0d8992e610c85",

"commissionTier": "silver"

}

}

}

**DELETE /agents/:id**

Deactivates an agent (soft delete).

**Response:**

{

"success": true,

"message": "Agent deactivated successfully"

}

**University Management**

**GET /universities**

Retrieves all universities.

**Query Parameters:**

* page (default: 1): Page number
* limit (default: 10): Items per page
* search: Search by name or country
* country: Filter by country
* status: Filter by status (active, inactive)
* sortBy: Field to sort by
* sortOrder: Sort order (asc, desc)

**Response:**

{

"success": true,

"data": {

"universities": [

{

"id": "60d21b4667d0d8992e610c70",

"name": "University of Cambridge",

"shortName": "Cambridge",

"country": "United Kingdom",

"city": "Cambridge",

"ranking": 5,

"programsCount": 150,

"logoUrl": "https://storage.intered.com/logos/cambridge.png",

"status": "active",

"createdAt": "2024-10-01T10:00:00Z"

},

// Additional universities...

],

"pagination": {

"page": 1,

"limit": 10,

"totalPages": 5,

"totalItems": 48

}

}

}

**GET /universities/:id**

Retrieves a specific university with detailed information.

**Response:**

{

"success": true,

"data": {

"university": {

"id": "60d21b4667d0d8992e610c70",

"name": "University of Cambridge",

"shortName": "Cambridge",

"country": "United Kingdom",

"city": "Cambridge",

"ranking": {

"world": 5,

"national": 2

},

"logoUrl": "https://storage.intered.com/logos/cambridge.png",

"website": "https://www.cam.ac.uk",

"description": "The University of Cambridge is a collegiate research university in Cambridge, United Kingdom. Founded in 1209, it is the world's third-oldest university in continuous operation.",

"admissionRequirements": {

"undergraduate": {

"academicRequirements": "A-Levels with AAA or equivalent",

"englishRequirements": {

"ielts": 7.0,

"toefl": 100

},

"additionalRequirements": "Admissions test for most courses"

},

"postgraduate": {

"academicRequirements": "2:1 honours degree or equivalent",

"englishRequirements": {

"ielts": 7.5,

"toefl": 110

},

"additionalRequirements": "Research proposal for research degrees"

}

},

"tuitionFees": {

"undergraduate": {

"domestic": 9250,

"international": 25000

},

"postgraduate": {

"domestic": 12000,

"international": 30000

},

"currency": "GBP"

},

"programsCount": 150,

"applicationDeadlines": {

"undergraduate": "2025-10-15",

"postgraduate": "2025-07-31"

},

"contacts": [

{

"name": "Admissions Office",

"email": "admissions@cam.ac.uk",

"phone": "+44123456789"

}

],

"agreementDetails": {

"startDate": "2025-01-01",

"endDate": "2027-12-31",

"commissionRates": {

"undergraduate": 5,

"postgraduate": 7

},

"documentId": "60d21b4667d0d8992e610c71"

},

"status": "active",

"createdAt": "2024-10-01T10:00:00Z",

"updatedAt": "2025-03-15T14:30:00Z"

}

}

}

**POST /universities**

Creates a new university.

**Request:**

{

"name": "Imperial College London",

"shortName": "Imperial",

"country": "United Kingdom",

"city": "London",

"ranking": {

"world": 8,

"national": 3

},

"website": "https://www.imperial.ac.uk",

"description": "Imperial College London is a public research university in London focusing on science, engineering, medicine, and business.",

"admissionRequirements": {

"undergraduate": {

"academicRequirements": "A-Levels with A\*AA or equivalent",

"englishRequirements": {

"ielts": 6.5,

"toefl": 92

}

},

"postgraduate": {

"academicRequirements": "2:1 honours degree or equivalent",

"englishRequirements": {

"ielts": 7.0,

"toefl": 100

}

}

},

"tuitionFees": {

"undergraduate": {

"domestic": 9250,

"international": 29000

},

"postgraduate": {

"domestic": 14000,

"international": 32000

},

"currency": "GBP"

},

"applicationDeadlines": {

"undergraduate": "2025-10-15",

"postgraduate": "2025-06-30"

},

"contacts": [

{

"name": "Admissions Office",

"email": "admissions@imperial.ac.uk",

"phone": "+44987654321"

}

]

}

**Response:**

{

"success": true,

"data": {

"success": true, "data": { "university": { "id": "60d21b4667d0d8992e610c72", "name": "Imperial College London", "shortName": "Imperial", "country": "United Kingdom", "city": "London", "createdAt": "2025-05-01T14:00:00Z" } } }

#### PUT /universities/:id

Updates a university.

\*\*Request:\*\*

```json

{

"ranking": {

"world": 7,

"national": 3

},

"description": "Updated description of Imperial College London...",

"tuitionFees": {

"undergraduate": {

"domestic": 9250,

"international": 32000

},

"postgraduate": {

"domestic": 15000,

"international": 35000

},

"currency": "GBP"

}

}

**Response:**

{

"success": true,

"data": {

"university": {

"id": "60d21b4667d0d8992e610c72",

"updatedAt": "2025-05-01T14:30:00Z"

}

}

}

**POST /universities/:id/logo**

Uploads a university logo.

**Request:** Form data with file field logo

**Response:**

{

"success": true,

"data": {

"logoUrl": "https://storage.intered.com/logos/imperial.png"

}

}

**DELETE /universities/:id**

Deactivates a university (soft delete).

**Response:**

{

"success": true,

"message": "University deactivated successfully"

}

**Program Management**

**GET /programs**

Retrieves all programs.

**Query Parameters:**

* page (default: 1): Page number
* limit (default: 10): Items per page
* search: Search by name
* universityId: Filter by university
* level: Filter by level (bachelor, master, phd, diploma, certificate)
* country: Filter by university country
* status: Filter by status (active, inactive)
* sortBy: Field to sort by
* sortOrder: Sort order (asc, desc)

**Response:**

{

"success": true,

"data": {

"programs": [

{

"id": "60d21b4667d0d8992e610d01",

"name": "Computer Science",

"level": "bachelor",

"universityId": "60d21b4667d0d8992e610c70",

"universityName": "University of Cambridge",

"country": "United Kingdom",

"duration": "3 years",

"tuitionFee": 25000,

"feeCurrency": "GBP",

"status": "active",

"createdAt": "2024-11-15T10:00:00Z"

},

// Additional programs...

],

"pagination": {

"page": 1,

"limit": 10,

"totalPages": 15,

"totalItems": 145

}

}

}

**GET /programs/:id**

Retrieves a specific program with detailed information.

**Response:**

{

"success": true,

"data": {

"program": {

"id": "60d21b4667d0d8992e610d01",

"name": "Computer Science",

"level": "bachelor",

"universityId": "60d21b4667d0d8992e610c70",

"university": {

"name": "University of Cambridge",

"shortName": "Cambridge",

"country": "United Kingdom",

"city": "Cambridge",

"logoUrl": "https://storage.intered.com/logos/cambridge.png"

},

"description": "The Computer Science course at Cambridge provides a mixture of theoretical knowledge and practical experience...",

"duration": "3 years",

"tuitionFee": 25000,

"feeCurrency": "GBP",

"requirements": {

"academicRequirements": "A-Levels with A\*A\*A including Mathematics",

"englishRequirements": {

"ielts": 7.0,

"toefl": 100

},

"additionalRequirements": "STEP Mathematics recommended"

},

"applicationDeadline": "2025-10-15",

"applicationFee": 75,

"applicationFeeCurrency": "GBP",

"startDates": ["2025-10-01"],

"careers": [

"Software Developer",

"Systems Analyst",

"Data Scientist",

"AI Researcher"

],

"admissionRate": 15,

"curriculumHighlights": [

"Algorithms",

"Artificial Intelligence",

"Computer Systems",

"Software Engineering"

],

"status": "active",

"createdAt": "2024-11-15T10:00:00Z",

"updatedAt": "2025-02-10T11:30:00Z"

}

}

}

**POST /programs**

Creates a new program.

**Request:**

{

"name": "Artificial Intelligence",

"level": "master",

"universityId": "60d21b4667d0d8992e610c70",

"description": "This Master's program focuses on the development of AI systems and their applications...",

"duration": "1 year",

"tuitionFee": 30000,

"feeCurrency": "GBP",

"requirements": {

"academicRequirements": "2:1 honours degree in Computer Science or related field",

"englishRequirements": {

"ielts": 7.0,

"toefl": 100

},

"additionalRequirements": "Strong mathematics background"

},

"applicationDeadline": "2025-07-31",

"applicationFee": 75,

"applicationFeeCurrency": "GBP",

"startDates": ["2025-10-01"],

"careers": [

"AI Engineer",

"Machine Learning Developer",

"Research Scientist",

"Data Scientist"

],

"curriculumHighlights": [

"Machine Learning",

"Natural Language Processing",

"Computer Vision",

"Reinforcement Learning"

]

}

**Response:**

{

"success": true,

"data": {

"program": {

"id": "60d21b4667d0d8992e610d02",

"name": "Artificial Intelligence",

"level": "master",

"universityId": "60d21b4667d0d8992e610c70",

"universityName": "University of Cambridge",

"createdAt": "2025-05-01T15:00:00Z"

}

}

}

**PUT /programs/:id**

Updates a program.

**Request:**

{

"description": "Updated description of the AI program...",

"tuitionFee": 32000,

"requirements": {

"academicRequirements": "2:1 honours degree in Computer Science, Mathematics, or related field",

"englishRequirements": {

"ielts": 7.0,

"toefl": 100

}

},

"applicationDeadline": "2025-08-15"

}

**Response:**

{

"success": true,

"data": {

"program": {

"id": "60d21b4667d0d8992e610d02",

"updatedAt": "2025-05-01T15:30:00Z"

}

}

}

**DELETE /programs/:id**

Deactivates a program (soft delete).

**Response:**

{

"success": true,

"message": "Program deactivated successfully"

}

**Application Management**

**GET /applications**

Retrieves all applications.

**Query Parameters:**

* page (default: 1): Page number
* limit (default: 10): Items per page
* search: Search by reference number or student name
* status: Filter by status (draft, submitted, under\_review, etc.)
* studentId: Filter by student
* agentId: Filter by agent
* universityId: Filter by university
* programId: Filter by program
* startDate: Filter by submission date (start)
* endDate: Filter by submission date (end)
* sortBy: Field to sort by
* sortOrder: Sort order (asc, desc)

**Response:**

{

"success": true,

"data": {

"applications": [

{

"id": "60d21b4667d0d8992e610d10",

"referenceNumber": "APP-2025-001",

"studentId": "60d21b4667d0d8992e610c86",

"studentName": "John Doe",

"universityId": "60d21b4667d0d8992e610c70",

"universityName": "University of Cambridge",

"programId": "60d21b4667d0d8992e610d01",

"programName": "Computer Science",

"status": "submitted",

"submittedAt": "2025-04-15T14:30:00Z",

"agentId": "60d21b4667d0d8992e610c95",

"agentName": "Global Education Agency",

"createdAt": "2025-04-10T09:00:00Z"

},

// Additional applications...

],

"pagination": {

"page": 1,

"limit": 10,

"totalPages": 6,

"totalItems": 58

}

}

}

**GET /applications/:id**

Retrieves a specific application with detailed information.

**Response:**

{

"success": true,

"data": {

"application": {

"id": "60d21b4667d0d8992e610d10",

"referenceNumber": "APP-2025-001",

"student": {

"id": "60d21b4667d0d8992e610c86",

"firstName": "John",

"lastName": "Doe",

"email": "john.doe@example.com",

"phone": "+1234567890"

},

"university": {

"id": "60d21b4667d0d8992e610c70",

"name": "University of Cambridge",

"country": "United Kingdom",

"logoUrl": "https://storage.intered.com/logos/cambridge.png"

},

"program": {

"id": "60d21b4667d0d8992e610d01",

"name": "Computer Science",

"level": "bachelor",

"duration": "3 years"

},

"agent": {

"id": "60d21b4667d0d8992e610c95",

"name": "Global Education Agency",

"email": "jane.smith@globaledu.com"

},

"status": "submitted",

"statusHistory": [

{

"status": "draft",

"date": "2025-04-10T09:00:00Z",

"updatedBy": "60d21b4667d0d8992e610c86"

},

{

"status": "submitted",

"date": "2025-04-15T14:30:00Z",

"updatedBy": "60d21b4667d0d8992e610c95"

}

],

"applicationData": {

"personalDetails": {

"firstName": "John",

"lastName": "Doe",

"dateOfBirth": "1998-05-15",

"nationality": "United States",

"passportNumber": "US123456789",

"address": {

"line1": "123 Main St",

"city": "Boston",

"state": "MA",

"postalCode": "02115",

"country": "United States"

}

},

"educationHistory": [

{

"institution": "Boston High School",

"qualification": "High School Diploma",

"fieldOfStudy": "General",

"startDate": "2014-09-01",

"endDate": "2018-06-30",

"grade": "A"

}

],

"testScores": [

{

"test": "IELTS",

"score": 7.5,

"dateObtained": "2024-11-15"

},

{

"test": "SAT",

"score": 1450,

"dateObtained": "2024-12-10"

}

],

"personalStatement": "I am applying to this program because...",

"references": [

{

"name": "Dr. Robert Brown",

"position": "Science Teacher",

"institution": "Boston High School",

"email": "r.brown@bostonhs.edu",

"phone": "+1234567888"

}

]

},

"documents": [

{

"id": "60d21b4667d0d8992e610d20",

"type": "transcript",

"name": "High School Transcript",

"status": "verified",

"uploadedAt": "2025-04-12T10:00:00Z",

"verifiedAt": "2025-04-16T09:30:00Z",

"verifiedBy": "60d21b4667d0d8992e610c85"

},

{

"id": "60d21b4667d0d8992e610d21",

"type": "passport",

"name": "Passport",

"status": "verified",

"uploadedAt": "2025-04-12T10:15:00Z",

"verifiedAt": "2025-04-16T09:35:00Z",

"verifiedBy": "60d21b4667d0d8992e610c85"

},

{

"id": "60d21b4667d0d8992e610d22",

"type": "ielts",

"name": "IELTS Certificate",

"status": "verified",

"uploadedAt": "2025-04-12T10:30:00Z",

"verifiedAt": "2025-04-16T09:40:00Z",

"verifiedBy": "60d21b4667d0d8992e610c85"

}

],

"applicationFee": {

"amount": 75,

"currency": "GBP",

"status": "paid",

"paidAt": "2025-04-14T15:30:00Z",

"paymentMethod": "credit\_card",

"transactionId": "pay\_12345678"

},

"offers": [],

"notes": [

{

"content": "Applicant has excellent academic background.",

"createdBy": "60d21b4667d0d8992e610c95",

"createdAt": "2025-04-15T14:00:00Z",

"isPrivate": true

}

],

"submittedAt": "2025-04-15T14:30:00Z",

"createdAt": "2025-04-10T09:00:00Z",

"updatedAt": "2025-04-15T14:30:00Z"

}

}

}

**POST /applications**

Creates a new application.

**Request:**

{

"studentId": "60d21b4667d0d8992e610c87",

"universityId": "60d21b4667d0d8992e610c70",

"programId": "60d21b4667d0d8992e610d01",

"agentId": "60d21b4667d0d8992e610c95",

"applicationData": {

"personalDetails": {

"firstName": "Jane",

"lastName": "Smith",

"dateOfBirth": "1999-03-20",

"nationality": "United Kingdom",

"passportNumber": "UK987654321",

"address": {

"line1": "456 High St",

"city": "London",

"postalCode": "E1 6AN",

"country": "United Kingdom"

}

},

"personalStatement": "I am applying to this program because I have a strong interest in computer science...",

"references": [

{

"name": "Prof. Sarah Williams",

"position": "Mathematics Teacher",

"institution": "London Academy",

"email": "s.williams@londonacademy.ac.uk",

"phone": "+4412345678"

}

]

},

"documentIds": ["60d21b4667d0d8992e610d23", "60d21b4667d0d8992e610d24"]

}

**Response:**

{

"success": true,

"data": {

"application": {

"id": "60d21b4667d0d8992e610d11",

"referenceNumber": "APP-2025-002",

"status": "draft",

"createdAt": "2025-05-01T16:00:00Z"

}

}

}

**PUT /applications/:id**

Updates an application.

**Request:**

{

"applicationData": {

"personalStatement": "Updated personal statement...",

"references": [

{

"name": "Prof. Sarah Williams",

"position": "Mathematics Teacher",

"institution": "London Academy",

"email": "s.williams@londonacademy.ac.uk",

"phone": "+4412345678"

},

{

"name": "Dr. James Johnson",

"position": "Computer Science Professor",

"institution": "London Tech Institute",

"email": "j.johnson@lti.ac.uk",

"phone": "+4487654321"

}

]

},

"documentIds": ["60d21b4667d0d8992e610d23", "60d21b4667d0d8992e610d24", "60d21b4667d0d8992e610d25"]

}

**Response:**

{

"success": true,

"data": {

"application": {

"id": "60d21b4667d0d8992e610d11",

"updatedAt": "2025-05-01T16:30:00Z"

}

}

}

**PUT /applications/:id/status**

Updates an application status.

**Request:**

{

"status": "submitted",

"notes": "Application submitted by student"

}

**Response:**

{

"success": true,

"data": {

"application": {

"id": "60d21b4667d0d8992e610d11",

"status": "submitted",

"submittedAt": "2025-05-01T17:00:00Z",

"updatedAt": "2025-05-01T17:00:00Z"

}

}

}

**POST /applications/:id/notes**

Adds a note to an application.

**Request:**

{

"content": "Student's academic credentials are impressive.",

"isPrivate": true

}

**Response:**

{

"success": true,

"data": {

"note": {

"id": "60d21b4667d0d8992e610d30",

"content": "Student's academic credentials are impressive.",

"createdBy": "60d21b4667d0d8992e610c85",

"createdAt": "2025-05-01T17:30:00Z",

"isPrivate": true

}

}

}

**POST /applications/:id/offers**

Adds an offer to an application.

**Request:**

{

"type": "conditional",

"offerDate": "2025-05-10T09:00:00Z",

"responseDeadline": "2025-06-10T23:59:59Z",

"conditions": [

{

"description": "Achieve a minimum of AAB in A-Levels including Mathematics at grade A"

},

{

"description": "Provide passport photocopy"

}

],

"documentId": "60d21b4667d0d8992e610d40"

}

**Response:**

{

"success": true,

"data": {

"offer": {

"id": "60d21b4667d0d8992e610d50",

"type": "conditional",

"offerDate": "2025-05-10T09:00:00Z",

"responseDeadline": "2025-06-10T23:59:59Z",

"createdAt": "2025-05-01T18:00:00Z"

}

}

}

**PUT /applications/:id/offers/:offerId/response**

Updates a student's response to an offer.

**Request:**

{

"status": "accepted",

"date": "2025-05-15T14:00:00Z",

"notes": "Student has accepted the offer"

}

**Response:**

{

"success": true,

"data": {

"offer": {

"id": "60d21b4667d0d8992e610d50",

"studentResponse": {

"status": "accepted",

"date": "2025-05-15T14:00:00Z"

},

"updatedAt": "2025-05-01T18:30:00Z"

}

}

}

**Document Management**

**GET /documents**

Retrieves all documents.

**Query Parameters:**

* page (default: 1): Page number
* limit (default: 10): Items per page
* ownerId: Filter by owner
* ownerType: Filter by owner type (student, agent, university)
* type: Filter by document type
* status: Filter by verification status (pending, verified, rejected)
* sortBy: Field to sort by
* sortOrder: Sort order (asc, desc)

**Response:**

{

"success": true,

"data": {

"documents": [

{

"id": "60d21b4667d0d8992e610d20",

"ownerId": "60d21b4667d0d8992e610c86",

"ownerType": "student",

"ownerName": "John Doe",

"type": "transcript",

"name": "High School Transcript",

"fileType": "application/pdf",

"fileSize": 1234567,

"status": "verified",

"uploadedAt": "2025-04-12T10:00:00Z",

"verifiedAt": "2025-04-16T09:30:00Z"

},

// Additional documents...

],

"pagination": {

"page": 1,

"limit": 10,

"totalPages": 10,

"totalItems": 95

}

}

}

**GET /documents/:id**

Retrieves a specific document with detailed information.

**Response:**

{

"success": true,

"data": {

"document": {

"id": "60d21b4667d0d8992e610d20",

"ownerId": "60d21b4667d0d8992e610c86",

"ownerType": "student",

"owner": {

"id": "60d21b4667d0d8992e610c86",

"firstName": "John",

"lastName": "Doe",

"email": "john.doe@example.com"

},

"relatedId": "60d21b4667d0d8992e610d10",

"relatedType": "application",

"type": "transcript",

"name": "High School Transcript",

"filePath": "students/60d21b4667d0d8992e610c86/transcript.pdf",

"fileUrl": "https://storage.intered.com/documents/students/60d21b4667d0d8992e610c86/transcript.pdf",

"fileType": "application/pdf",

"fileSize": 1234567,

"status": "verified",

"statusHistory": [

{

"status": "pending",

"date": "2025-04-12T10:00:00Z",

"updatedBy": "60d21b4667d0d8992e610c86"

},

{

"status": "verified",

"date": "2025-04-16T09:30:00Z",

"updatedBy": "60d21b4667d0d8992e610c85",

"comments": "Document verified successfully"

}

],

"verifiedBy": {

"id": "60d21b4667d0d8992e610c85",

"firstName": "Admin",

"lastName": "User"

},

"verifiedAt": "2025-04-16T09:30:00Z",

"metadata": {

"issueDate": "2018-06-30",

"issuedBy": "Boston High School"

},

"uploadedBy": "60d21b4667d0d8992e610c86",

"uploadedAt": "2025-04-12T10:00:00Z",

"updatedAt": "2025-04-16T09:30:00Z"

}

}

}

**POST /documents**

Uploads a new document.

**Request:** Form data with the following fields:

* file: File to upload
* ownerId: Document owner ID
* ownerType: Document owner type (student, agent, university)
* relatedId (optional): Related entity ID (e.g., application)
* relatedType (optional): Related entity type (e.g., application)
* type: Document type
* name: Document name
* metadata (optional): Additional metadata as JSON string

**Response:**

{

"success": true,

"data": {

"document": {

"id": "60d21b4667d0d8992e610d26",

"ownerId": "60d21b4667d0d8992e610c87",

"ownerType": "student",

"type": "passport",

"name": "Passport",

"fileUrl": "https://storage.intered.com/documents/students/60d21b4667d0d8992e610c87/passport.pdf",

"status": "pending",

"uploadedAt": "2025-05-01T19:00:00Z"

}

}

}

**PUT /documents/:id/verify**

Verifies a document.

**Request:**

{

"status": "verified",

"comments": "Document verified successfully"

}

**Response:**

{

"success": true,

"data": {

"document": {

"id": "60d21b4667d0d8992e610d26",

"status": "verified",

"verifiedBy": "60d21b4667d0d8992e610c85",

"verifiedAt": "2025-05-01T19:30:00Z",

"updatedAt": "2025-05-01T19:30:00Z"

}

}

}

**DELETE /documents/:id**

Deletes a document.

**Response:**

{

"success": true,

"message": "Document deleted successfully"

}

**Commission Management**

**GET /commissions**

Retrieves all commissions.

**Query Parameters:**

* page (default: 1): Page number
* limit (default: 10): Items per page
* agentId: Filter by agent
* status: Filter by status (pending, approved, paid, rejected)
* startDate: Filter by date (start)
* endDate: Filter by date (end)
* sortBy: Field to sort by
* sortOrder: Sort order (asc, desc)

**Response:**

{

"success": true,

"data": {

"commissions": [

{

"id": "60d21b4667d0d8992e610e01",

"agentId": "60d21b4667d0d8992e610c95",

"agentName": "Global Education Agency",

"applicationId": "60d21b4667d0d8992e610d10",

"studentName": "John Doe",

"universityName": "University of Cambridge",

"programName": "Computer Science",

"amount": 1250,

"currency": "GBP",

"status": "approved",

"createdAt": "2025-04-20T10:00:00Z"

},

// Additional commissions...

],

"pagination": {

"page": 1,

"limit": 10,

"totalPages": 3,

"totalItems": 28

}

}

}

**GET /commissions/:id**

Retrieves a specific commission with detailed information.

**Response:**

{

"success": true,

"data": {

"commission": {

"id": "60d21b4667d0d8992e610e01",

"agent": {

"id": "60d21b4667d0d8992e610c95",

"name": "Global Education Agency",

"email": "jane.smith@globaledu.com"

},

"application": {

"id": "60d21b4667d0d8992e610d10",

"referenceNumber": "APP-2025-001"

},

"student": {

"id": "60d21b4667d0d8992e610c86",

"firstName": "John",

"lastName": "Doe"

},

"university": {

"id": "60d21b4667d0d8992e610c70",

"name": "University of Cambridge"

},

"program": {

"id": "60d21b4667d0d8992e610d01",

"name": "Computer Science",

"level": "bachelor"

},

"amount": 1250,

"currency": "GBP",

"rate": 5,

"status": "approved",

"statusHistory": [

{

"status": "pending",

"date": "2025-04-20T10:00:00Z"

},

{

"status": "approved",

"date": "2025-04-25T14:30:00Z",

"updatedBy": "60d21b4667d0d8992e610c85"

}

],

"approvalDetails": {

"appro# InterEd Admin Portal API Specification

## Overview

This document provides a comprehensive specification for the InterEd Student Recruitment Admin Portal API. It defines all endpoints, request/response formats, authentication mechanisms, and error handling patterns necessary for implementing the backend services for the admin portal.

The API is designed following RESTful principles with JWT authentication and is intended to be implemented using Express.js and integrated with Supabase.

## Base URL

https://api.intered.com/v1

For development environments:

http://localhost:5000/api/v1

## Authentication

### Authentication Mechanism

The API uses JSON Web Tokens (JWT) for authentication:

1. All authenticated requests must include an `Authorization` header with a Bearer token

2. Tokens expire after 1 hour

3. Refresh tokens are used for obtaining new access tokens

4. Role-based permissions control access to resources

### Authentication Endpoints

#### POST /auth/login

Authenticates a user and returns a JWT token.

\*\*Request:\*\*

```json

{

"email": "admin@intered.com",

"password": "securepassword"

}

**Response:**

{

"success": true,

"data": {

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",

"refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",

"expiresIn": 3600,

"user": {

"id": "60d21b4667d0d8992e610c85",

"email": "admin@intered.com",

"firstName": "Admin",

"lastName": "User",

"role": "admin"

}

}

}

**POST /auth/refresh**

Refreshes an expired JWT token.

**Request:**

{

"refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."

}

**Response:**

{

"success": true,

"data": {

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",

"refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",

"expiresIn": 3600

}

}

**POST /auth/register**

Registers a new user (admin-only operation).

**Request:**

{

"email": "staff@intered.com",

"password": "securepassword",

"firstName": "Staff",

"lastName": "User",

"role": "staff"

}

**Response:**

{

"success": true,

"data": {

"user": {

"id": "60d21b4667d0d8992e610c85",

"email": "staff@intered.com",

"firstName": "Staff",

"lastName": "User",

"role": "staff"

}

}

}

**POST /auth/forgot-password**

Initiates password reset process.

**Request:**

{

"email": "admin@intered.com"

}

**Response:**

{

"success": true,

"message": "Password reset instructions sent to email"

}

**POST /auth/reset-password**

Resets password using reset token.

**Request:**

{

"token": "reset-token-from-email",

"newPassword": "newsecurepassword"

}

**Response:**

{

"success": true,

"message": "Password reset successfully"

}

**POST /auth/logout**

Invalidates the current JWT token.

**Response:**

{

"success": true,

"message": "Logged out successfully"

}

**Core API Endpoints**

**Users and Profile Management**

**GET /users**

Retrieves all users (admin-only).

**Query Parameters:**

* page (default: 1): Page number
* limit (default: 10): Items per page
* role: Filter by role (admin, staff, agent, student)
* search: Search by name or email
* sortBy: Field to sort by
* sortOrder: Sort order (asc, desc)

**Response:**

{

"success": true,

"data": {

"users": [

{

"id": "60d21b4667d0d8992e610c85",

"email": "admin@intered.com",

"firstName": "Admin",

"lastName": "User",

"role": "admin",

"lastLogin": "2025-05-01T10:30:00Z",

"status": "active",

"createdAt": "2025-01-01T10:00:00Z"

},

// Additional users...

],

"pagination": {

"page": 1,

"limit": 10,

"totalPages": 5,

"totalItems": 45

}

}

}

**GET /users/:id**

Retrieves a specific user.

**Response:**

{

"success": true,

"data": {

"user": {

"id": "60d21b4667d0d8992e610c85",

"email": "admin@intered.com",

"firstName": "Admin",

"lastName": "User",

"role": "admin",

"phone": "+1234567890",

"avatar": "https://storage.intered.com/avatars/admin.jpg",

"lastLogin": "2025-05-01T10:30:00Z",

"status": "active",

"createdAt": "2025-01-01T10:00:00Z",

"updatedAt": "2025-05-01T10:30:00Z"

}

}

}

**PUT /users/:id**

Updates a user.

**Request:**

{

"firstName": "Updated",

"lastName": "Name",

"phone": "+1234567890",

"status": "active"

}

**Response:**

{

"success": true,

"data": {

"user": {

"id": "60d21b4667d0d8992e610c85",

"firstName": "Updated",

"lastName": "Name",

"phone": "+1234567890",

"status": "active",

"updatedAt": "2025-05-01T10:35:00Z"

}

}

}

**DELETE /users/:id**

Deactivates a user (soft delete).

**Response:**

{

"success": true,

"message": "User deactivated successfully"

}

**POST /users/:id/avatar**

Uploads a user avatar.

**Request:** Form data with file field avatar

**Response:**

{

"success": true,

"data": {

"avatarUrl": "https://storage.intered.com/avatars/admin.jpg"

}

}

**Student Management**

**GET /students**

Retrieves all students.

**Query Parameters:**

* page (default: 1): Page number
* limit (default: 10): Items per page
* search: Search by name or email
* status: Filter by status (active, inactive, graduated, withdrawn)
* agentId: Filter by agent
* sortBy: Field to sort by
* sortOrder: Sort order (asc, desc)

**Response:**

{

"success": true,

"data": {

"students": [

{

"id": "60d21b4667d0d8992e610c86",

"profileId": "60d21b4667d0d8992e610c90",

"firstName": "John",

"lastName": "Doe",

"email": "john.doe@example.com",

"phone": "+1234567890",

"agentId": "60d21b4667d0d8992e610c95",

"agentName": "Global Education Agency",

"status": "active",

"applicationCount": 2,

"createdAt": "2025-01-15T10:00:00Z"

},

// Additional students...

],

"pagination": {

"page": 1,

"limit": 10,

"totalPages": 8,

"totalItems": 76

}

}

}

**GET /students/:id**

Retrieves a specific student with detailed information.

**Response:**

{

"success": true,

"data": {

"student": {

"id": "60d21b4667d0d8992e610c86",

"profileId": "60d21b4667d0d8992e610c90",

"firstName": "John",

"lastName": "Doe",

"email": "john.doe@example.com",

"phone": "+1234567890",

"dateOfBirth": "1998-05-15",

"nationality": "United States",

"address": {

"line1": "123 Main St",

"line2": "Apt 4B",

"city": "Boston",

"state": "MA",

"postalCode": "02115",

"country": "United States"

},

"academicInfo": {

"highestEducation": "High School",

"gpa": 3.8,

"institutions": [

{

"name": "Boston High School",

"degree": "High School Diploma",

"fieldOfStudy": "General",

"startDate": "2014-09-01",

"endDate": "2018-06-30",

"grade": "A"

}

]

},

"testScores": {

"ielts": {

"overall": 7.5,

"listening": 7.5,

"reading": 8.0,

"writing": 7.0,

"speaking": 7.5,

"testDate": "2024-11-15"

}

},

"preferences": {

"destinations": ["UK", "Canada", "Australia"],

"programLevels": ["Undergraduate"],

"fieldOfStudy": ["Computer Science", "Information Technology"],

"budget": {

"amount": 30000,

"currency": "USD"

}

},

"agent": {

"id": "60d21b4667d0d8992e610c95",

"name": "Global Education Agency",

"counselorName": "Jane Smith",

"email": "jane.smith@globaledu.com"

},

"status": "active",

"createdAt": "2025-01-15T10:00:00Z",

"updatedAt": "2025-04-20T14:30:00Z"

}

}

}

**POST /students**

Creates a new student.

**Request:**

{

"firstName": "Jane",

"lastName": "Smith",

"email": "jane.smith@example.com",

"phone": "+1987654321",

"dateOfBirth": "1999-03-20",

"nationality": "United Kingdom",

"address": {

"line1": "456 High St",

"city": "London",

"postalCode": "E1 6AN",

"country": "United Kingdom"

},

"academicInfo": {

"highestEducation": "High School",

"gpa": 4.0,

"institutions": [

{

"name": "London Academy",

"degree": "A-Levels",

"fieldOfStudy": "Sciences",

"startDate": "2016-09-01",

"endDate": "2018-06-30",

"grade": "A\*"

}

]

},

"agentId": "60d21b4667d0d8992e610c95"

}

**Response:**

{

"success": true,

"data": {

"student": {

"id": "60d21b4667d0d8992e610c87",

"email": "jane.smith@example.com",

"firstName": "Jane",

"lastName": "Smith",

"createdAt": "2025-05-01T11:00:00Z"

}

}

}

**PUT /students/:id**

Updates a student.

**Request:**

{

"phone": "+1987654322",

"address": {

"line1": "789 New Ave",

"city": "London",

"postalCode": "E2 8LP",

"country": "United Kingdom"

},

"preferences": {

"destinations": ["UK", "Canada"],

"programLevels": ["Undergraduate"],

"fieldOfStudy": ["Computer Science"]

}

}

**Response:**

{

"success": true,

"data": {

"student": {

"id": "60d21b4667d0d8992e610c87",

"updatedAt": "2025-05-01T11:30:00Z"

}

}

}

**DELETE /students/:id**

Deactivates a student (soft delete).

**Response:**

{

"success": true,

"message": "Student deactivated successfully"

}

**Agent Management**

**GET /agents**

Retrieves all agents.

**Query Parameters:**

* page (default: 1): Page number
* limit (default: 10): Items per page
* search: Search by name, agency name, or email
* status: Filter by status (active, inactive, suspended)
* verificationStatus: Filter by verification status (pending, verified, rejected)
* sortBy: Field to sort by
* sortOrder: Sort order (asc, desc)

**Response:**

{

"success": true,

"data": {

"agents": [

{

"id": "60d21b4667d0d8992e610c95",

"profileId": "60d21b4667d0d8992e610c96",

"firstName": "Jane",

"lastName": "Smith",

"email": "jane.smith@globaledu.com",

"phone": "+1234567890",

"agencyName": "Global Education Agency",

"agencyType": "company",

"verificationStatus": "verified",

"status": "active",

"studentCount": 15,

"applicationCount": 28,

"createdAt": "2024-12-01T10:00:00Z"

},

// Additional agents...

],

"pagination": {

"page": 1,

"limit": 10,

"totalPages": 3,

"totalItems": 25

}

}

}

**GET /agents/:id**

Retrieves a specific agent with detailed information.

**Response:**

{

"success": true,

"data": {

"agent": {

"id": "60d21b4667d0d8992e610c95",

"profileId": "60d21b4667d0d8992e610c96",

"firstName": "Jane",

"lastName": "Smith",

"email": "jane.smith@globaledu.com",

"phone": "+1234567890",

"agencyName": "Global Education Agency",

"agencyType": "company",

"businessInfo": {

"registrationNumber": "GEA12345",

"foundingYear": 2015,

"website": "https://www.globaledu.com",

"address": {

"line1": "100 Education St",

"city": "New York",

"state": "NY",

"postalCode": "10001",

"country": "United States"

}

},

"specializations": {

"countries": ["UK", "USA", "Canada", "Australia"],

"programLevels": ["Undergraduate", "Postgraduate"],

"disciplines": ["Business", "Engineering", "Computer Science"]

},

"verificationStatus": "verified",

"verificationDate": "2025-01-10T09:30:00Z",

"verifiedBy": "60d21b4667d0d8992e610c85",

"status": "active",

"commissionTier": "gold",

"parentAgentId": null,

"subAgents": [

{

"id": "60d21b4667d0d8992e610c97",

"name": "John Johnson",

"email": "john.johnson@globaledu.com",

"studentCount": 8

}

],

"performance": {

"studentCount": 15,

"applicationCount": 28,

"conversionRate": 85,

"commissionEarned": {

"amount": 25000,

"currency": "USD"

}

},

"agreementDetails": {

"startDate": "2025-01-01",

"endDate": "2025-12-31",

"documentId": "60d21b4667d0d8992e610c98"

},

"createdAt": "2024-12-01T10:00:00Z",

"updatedAt": "2025-04-15T15:30:00Z"

}

}

}

**POST /agents**

Creates a new agent.

**Request:**

{

"firstName": "Robert",

"lastName": "Johnson",

"email": "robert.johnson@studyabroad.com",

"phone": "+1122334455",

"agencyName": "Study Abroad Experts",

"agencyType": "company",

"businessInfo": {

"registrationNumber": "SAE54321",

"foundingYear": 2018,

"website": "https://www.studyabroad.com",

"address": {

"line1": "200 Education Blvd",

"city": "Chicago",

"state": "IL",

"postalCode": "60601",

"country": "United States"

}

},

"specializations": {

"countries": ["USA", "Canada"],

"programLevels": ["Undergraduate", "Postgraduate"],

"disciplines": ["Business", "Arts"]

}

}

**Response:**

{

"success": true,

"data": {

"agent": {

"id": "60d21b4667d0d8992e610c99",

"email": "robert.johnson@studyabroad.com",

"firstName": "Robert",

"lastName": "Johnson",

"agencyName": "Study Abroad Experts",

"verificationStatus": "pending",

"createdAt": "2025-05-01T12:30:00Z"

}

}

}

**PUT /agents/:id**

Updates an agent.

**Request:**

{

"phone": "+1122334466",

"businessInfo": {

"website": "https://www.studyabroadexperts.com",

"address": {

"line1": "300 Education Ctr",

"city": "Chicago",

"state": "IL",

"postalCode": "60602",

"country": "United States"

}

},

"specializations": {

"countries": ["USA", "Canada", "UK"],

"programLevels": ["Undergraduate", "Postgraduate", "Diploma"],

"disciplines": ["Business", "Arts", "Technology"]

}

}

**Response:**

{

"success": true,

"data": {

"agent": {

"id": "60d21b4667d0d8992e610c99",

"updatedAt": "2025-05-01T13:00:00Z"

}

}

}

**PUT /agents/:id/verify**

Verifies an agent (admin/staff only).

**Request:**

{

"verificationStatus": "verified",

"commissionTier": "silver",

"notes": "Documents verified successfully"

}

**Response:**

{

"success": true,

"data": {

"agent": {

"id": "60d21b4667d0d8992e610c99",

"verificationStatus": "verified",

"verificationDate": "2025-05-01T13:30:00Z",

"verifiedBy": "60d21b4667d0d8992e610c85",

"commissionTier": "silver"

}

}

}

**DELETE /agents/:id**

Deactivates an agent (soft delete).

**Response:**

{

"success": true,

"message": "Agent deactivated successfully"

}

**University Management**

**GET /universities**

Retrieves all universities.

**Query Parameters:**

* page (default: 1): Page number
* limit (default: 10): Items per page
* search: Search by name or country
* country: Filter by country
* status: Filter by status (active, inactive)
* sortBy: Field to sort by
* sortOrder: Sort order (asc, desc)

**Response:**

{

"success": true,

"data": {

"universities": [

{

"id": "60d21b4667d0d8992e610c70",

"name": "University of Cambridge",

"shortName": "Cambridge",

"country": "United Kingdom",

"city": "Cambridge",

"ranking": 5,

"programsCount": 150,

"logoUrl": "https://storage.intered.com/logos/cambridge.png",

"status": "active",

"createdAt": "2024-10-01T10:00:00Z"

},

// Additional universities...

],

"pagination": {

"page": 1,

"limit": 10,

"totalPages": 5,

"totalItems": 48

}

}

}

**GET /universities/:id**

Retrieves a specific university with detailed information.

**Response:**

{

"success": true,

"data": {

"university": {

"id": "60d21b4667d0d8992e610c70",

"name": "University of Cambridge",

"shortName": "Cambridge",

"country": "United Kingdom",

"city": "Cambridge",

"ranking": {

"world": 5,

"national": 2

},

"logoUrl": "https://storage.intered.com/logos/cambridge.png",

"website": "https://www.cam.ac.uk",

"description": "The University of Cambridge is a collegiate research university in Cambridge, United Kingdom. Founded in 1209, it is the world's third-oldest university in continuous operation.",

"admissionRequirements": {

"undergraduate": {

"academicRequirements": "A-Levels with AAA or equivalent",

"englishRequirements": {

"ielts": 7.0,

"toefl": 100

},

"additionalRequirements": "Admissions test for most courses"

},

"postgraduate": {

"academicRequirements": "2:1 honours degree or equivalent",

"englishRequirements": {

"ielts": 7.5,

"toefl": 110

},

"additionalRequirements": "Research proposal for research degrees"

}

},

"tuitionFees": {

"undergraduate": {

"domestic": 9250,

"international": 25000

},

"postgraduate": {

"domestic": 12000,

"international": 30000

},

"currency": "GBP"

},

"programsCount": 150,

"applicationDeadlines": {

"undergraduate": "2025-10-15",

"postgraduate": "2025-07-31"

},

"contacts": [

{

"name": "Admissions Office",

"email": "admissions@cam.ac.uk",

"phone": "+44123456789"

}

],

"agreementDetails": {

"startDate": "2025-01-01",

"endDate": "2027-12-31",

"commissionRates": {

"undergraduate": 5,

"postgraduate": 7

},

"documentId": "60d21b4667d0d8992e610c71"

},

"status": "active",

"createdAt": "2024-10-01T10:00:00Z",

"updatedAt": "2025-03-15T14:30:00Z"

}

}

}

**POST /universities**

Creates a new university.

**Request:**

{

"name": "Imperial College London",

"shortName": "Imperial",

"country": "United Kingdom",

"city": "London",

"ranking": {

"world": 8,

"national": 3

},

"website": "https://www.imperial.ac.uk",

"description": "Imperial College London is a public research university in London focusing on science, engineering, medicine, and business.",

"admissionRequirements": {

"undergraduate": {

"academicRequirements": "A-Levels with A\*AA or equivalent",

"englishRequirements": {

"ielts": 6.5,

"toefl": 92

}

},

"postgraduate": {

"academicRequirements": "2:1 honours degree or equivalent",

"englishRequirements": {

"ielts": 7.0,

"toefl": 100

}

}

},

"tuitionFees": {

"undergraduate": {

"domestic": 9250,

"international": 29000

},

"postgraduate": {

"domestic": 14000,

"international": 32000

},

"currency": "GBP"

},

"applicationDeadlines": {

"undergraduate": "2025-10-15",

"postgraduate": "2025-06-30"

},

"contacts": [

{

"name": "Admissions Office",

"email": "admissions@imperial.ac.uk",

"phone": "+44987654321"

}

]

}

**Response:**

{

"success": true,

"data": {

"approvalDetails": { "approvedBy": "60d21b4667d0d8992e610c85", "approvedAt": "2025-04-25T14:30:00Z", "comments": "Commission approved as per standard rate" }, "payment": null, "createdAt": "2025-04-20T10:00:00Z", "updatedAt": "2025-04-25T14:30:00Z" } } }

#### POST /commissions

Creates a new commission.

\*\*Request:\*\*

```json

{

"agentId": "60d21b4667d0d8992e610c95",

"applicationId": "60d21b4667d0d8992e610d11",

"amount": 1600,

"currency": "GBP",

"rate": 5,

"notes": "Commission for successful enrollment"

}

**Response:**

{

"success": true,

"data": {

"commission": {

"id": "60d21b4667d0d8992e610e02",

"agentId": "60d21b4667d0d8992e610c95",

"applicationId": "60d21b4667d0d8992e610d11",

"amount": 1600,

"currency": "GBP",

"status": "pending",

"createdAt": "2025-05-01T20:00:00Z"

}

}

}

**PUT /commissions/:id/approve**

Approves a commission.

**Request:**

{

"comments": "Commission approved as per master agent agreement"

}

**Response:**

{

"success": true,

"data": {

"commission": {

"id": "60d21b4667d0d8992e610e02",

"status": "approved",

"approvalDetails": {

"approvedBy": "60d21b4667d0d8992e610c85",

"approvedAt": "2025-05-01T20:30:00Z",

"comments": "Commission approved as per master agent agreement"

},

"updatedAt": "2025-05-01T20:30:00Z"

}

}

}

**PUT /commissions/:id/payment**

Marks a commission as paid.

**Request:**

{

"paymentDate": "2025-05-01T21:00:00Z",

"method": "bank\_transfer",

"reference": "TRX12345678",

"notes": "Payment processed"

}

**Response:**

{

"success": true,

"data": {

"commission": {

"id": "60d21b4667d0d8992e610e02",

"status": "paid",

"payment": {

"paymentDate": "2025-05-01T21:00:00Z",

"method": "bank\_transfer",

"reference": "TRX12345678"

},

"updatedAt": "2025-05-01T21:00:00Z"

}

}

}

**Dashboard and Analytics**

**GET /dashboard/stats**

Retrieves dashboard statistics.

**Response:**

{

"success": true,

"data": {

"stats": {

"students": {

"total": 1245,

"active": 987,

"newLast30Days": 118

},

"applications": {

"total": 2156,

"pending": 548,

"approved": 986,

"rejected": 345,

"withdrawn": 277

},

"agents": {

"total": 87,

"active": 75,

"pending": 12

},

"universities": {

"total": 48,

"programs": 723

},

"revenue": {

"totalApplicationFees": 162450,

"totalCommissionsPaid": 458000,

"currency": "USD"

}

}

}

}

**GET /dashboard/recent-applications**

Retrieves recent applications for dashboard.

**Query Parameters:**

* limit (default: 5): Number of applications to retrieve

**Response:**

{

"success": true,

"data": {

"applications": [

{

"id": "60d21b4667d0d8992e610d15",

"referenceNumber": "APP-2025-006",

"student": {

"id": "60d21b4667d0d8992e610c88",

"name": "Robert Wilson",

"email": "robert.wilson@example.com",

"avatarUrl": "https://storage.intered.com/avatars/robert.jpg"

},

"program": {

"name": "Business Administration",

"university": "Imperial College London"

},

"status": "submitted",

"submittedAt": "2025-05-01T09:30:00Z",

"agent": {

"name": "Study Abroad Experts"

}

},

// Additional applications...

]

}

}

**GET /dashboard/agent-performance**

Retrieves agent performance metrics for dashboard.

**Query Parameters:**

* period (default: "month"): Period for statistics (week, month, quarter, year)
* limit (default: 5): Number of agents to retrieve

**Response:**

{

"success": true,

"data": {

"agents": [

{

"id": "60d21b4667d0d8992e610c95",

"name": "Global Education Agency",

"applications": 28,

"approvals": 22,

"conversionRate": 78.6,

"commissions": {

"amount": 35000,

"currency": "USD"

}

},

// Additional agents...

],

"period": {

"start": "2025-04-01T00:00:00Z",

"end": "2025-04-30T23:59:59Z"

}

}

}

**GET /analytics/applications**

Retrieves application analytics.

**Query Parameters:**

* startDate: Start date for analysis
* endDate: End date for analysis
* groupBy (default: "month"): Grouping period (day, week, month, quarter, year)
* filters: JSON object with filter criteria

**Response:**

{

"success": true,

"data": {

"timeline": [

{

"period": "2025-01",

"submitted": 145,

"approved": 98,

"rejected": 32,

"withdrawn": 15

},

{

"period": "2025-02",

"submitted": 162,

"approved": 110,

"rejected": 35,

"withdrawn": 17

},

{

"period": "2025-03",

"submitted": 178,

"approved": 121,

"rejected": 38,

"withdrawn": 19

},

{

"period": "2025-04",

"submitted": 195,

"approved": 103,

"rejected": 25,

"withdrawn": 12

}

],

"byStatus": {

"submitted": 680,

"under\_review": 135,

"approved": 432,

"rejected": 130,

"withdrawn": 63

},

"byUniversity": [

{

"university": "University of Cambridge",

"count": 189

},

{

"university": "Imperial College London",

"count": 157

},

// Additional universities...

],

"byProgram": [

{

"program": "Computer Science",

"count": 87

},

{

"university": "Business Administration",

"count": 75

},

// Additional programs...

],

"conversionRate": 63.5,

"averageProcessingTime": 12 // days

}

}

**GET /analytics/students**

Retrieves student analytics.

**Query Parameters:** Similar to /analytics/applications

**Response:**

{

"success": true,

"data": {

"timeline": [

{

"period": "2025-01",

"registrations": 95

},

// Additional periods...

],

"byCountry": [

{

"country": "United States",

"count": 245

},

// Additional countries...

],

"byAge": [

{

"range": "18-20",

"count": 320

},

// Additional age ranges...

],

"byEducationLevel": [

{

"level": "High School",

"count": 450

},

// Additional levels...

],

"byReferralSource": [

{

"source": "Agent",

"count": 567

},

// Additional sources...

]

}

}

**Error Handling**

**Standard Error Response Format**

{

"success": false,

"error": {

"code": "ERROR\_CODE",

"message": "Human-readable error message",

"details": {} // Optional additional error information

}

}

**Common Error Codes**

| **Code** | **HTTP Status** | **Description** |
| --- | --- | --- |
| AUTHENTICATION\_ERROR | 401 | Authentication failed or token expired |
| AUTHORIZATION\_ERROR | 403 | User lacks permission for the requested action |
| RESOURCE\_NOT\_FOUND | 404 | The requested resource does not exist |
| VALIDATION\_ERROR | 422 | Request data failed validation |
| DUPLICATE\_RESOURCE | 409 | Resource already exists (e.g., email already registered) |
| SERVER\_ERROR | 500 | Unexpected server error |
| RATE\_LIMIT\_EXCEEDED | 429 | Too many requests |
| REQUIRED\_FIELD\_MISSING | 400 | Required field is missing in the request |
| INVALID\_FIELD\_FORMAT | 400 | Field has an invalid format |
| INVALID\_DATE\_RANGE | 400 | Date range is invalid |

**Validation Error Example**

{

"success": false,

"error": {

"code": "VALIDATION\_ERROR",

"message": "Validation failed",

"details": {

"fields": {

"email": "Must be a valid email address",

"password": "Must be at least 8 characters long"

}

}

}

}

**Pagination**

All endpoints that return lists support pagination with the following query parameters:

* page: Page number (default: 1)
* limit: Number of items per page (default: 10, max: 100)

The response includes pagination information:

{

"success": true,

"data": {

"items": [...],

"pagination": {

"page": 1,

"limit": 10,

"totalPages": 5,

"totalItems": 48

}

}

}

**Filtering and Sorting**

Endpoints support filtering via query parameters:

* Simple filters: /api/v1/students?status=active&country=US
* Date ranges: /api/v1/applications?submittedAt[gte]=2025-01-01&submittedAt[lte]=2025-03-31
* Sorting: /api/v1/students?sortBy=createdAt&sortOrder=desc

**API Versioning**

The API uses URL versioning:

* Current version: /api/v1/...
* Future versions: /api/v2/...

**Rate Limiting**

The API implements rate limiting to prevent abuse:

* 100 requests per minute for regular users
* 300 requests per minute for admin users

Rate limit headers are included in all responses:

X-RateLimit-Limit: 100

X-RateLimit-Remaining: 95

X-RateLimit-Reset: 1712345678

**Webhook Events**

The API can send webhook notifications for the following events:

| **Event** | **Description** |
| --- | --- |
| student.created | A new student is registered |
| student.updated | A student profile is updated |
| application.submitted | An application is submitted |
| application.status\_changed | An application status changes |
| document.uploaded | A document is uploaded |
| document.verified | A document is verified |
| offer.created | An offer is made to a student |
| offer.responded | A student responds to an offer |
| commission.approved | A commission is approved |
| commission.paid | A commission is paid |

Webhook payload format:

{

"id": "evt\_12345",

"type": "application.status\_changed",

"created": "2025-05-01T14:30:00Z",

"data": {

"applicationId": "60d21b4667d0d8992e610d10",

"previousStatus": "submitted",

"newStatus": "under\_review",

"timestamp": "2025-05-01T14:30:00Z",

"actor": "60d21b4667d0d8992e610c85"

}

}

**Implementation Notes**

**Backend Technology**

* Node.js with Express.js
* MongoDB with Mongoose ODM
* JWT for authentication
* SendGrid for email notifications
* Socket.io for real-time updates
* Supabase for storage and real-time features
* Redis for caching

**Security Considerations**

* All endpoints require HTTPS
* JWT tokens expire after 1 hour
* Passwords are hashed using bcrypt
* Rate limiting prevents brute force attacks
* Input validation prevents injection attacks
* CORS configuration limits allowed origins
* Sensitive data is encrypted in the database

**Conclusion**

This API specification provides a comprehensive foundation for implementing the InterEd Student Recruitment Admin Portal backend. By following these guidelines, developers can create a secure, scalable, and feature-rich API that supports all the functionality required by the admin portal.

The API is designed to be:

1. **RESTful** - Following REST principles for resource-oriented design
2. **Secure** - Implementing best practices for authentication and data protection
3. **Scalable** - Supporting growth in users and data volume
4. **Documented** - Providing clear specifications for all endpoints
5. **Consistent** - Maintaining a uniform approach to request/response formats
6. **Extensible** - Allowing for future features and integrations

This specification covers approximately 80% of the API requirements for the InterEd Admin Portal, providing a solid foundation for the Vibathon implementation. "success": true, "data": { "university": { "id": "60d21b4667d0d8992e610c72", "name": "Imperial College London", "shortName": "Imperial", "country": "United Kingdom", "city": "London", "createdAt": "2025-05-01T14:00:00Z" } } }

#### PUT /universities/:id

Updates a university.

\*\*Request:\*\*

```json

{

"ranking": {

"world": 7,

"national": 3

},

"description": "Updated description of Imperial College London...",

"tuitionFees": {

"undergraduate": {

"domestic": 9250,

"international": 32000

},

"postgraduate": {

"domestic": 15000,

"international": 35000

},

"currency": "GBP"

}

}

**Response:**

{

"success": true,

"data": {

"university": {

"id": "60d21b4667d0d8992e610c72",

"updatedAt": "2025-05-01T14:30:00Z"

}

}

}

**POST /universities/:id/logo**

Uploads a university logo.

**Request:** Form data with file field logo

**Response:**

{

"success": true,

"data": {

"logoUrl": "https://storage.intered.com/logos/imperial.png"

}

}

**DELETE /universities/:id**

Deactivates a university (soft delete).

**Response:**

{

"success": true,

"message": "University deactivated successfully"

}

**Program Management**

**GET /programs**

Retrieves all programs.

**Query Parameters:**

* page (default: 1): Page number
* limit (default: 10): Items per page
* search: Search by name
* universityId: Filter by university
* level: Filter by level (bachelor, master, phd, diploma, certificate)
* country: Filter by university country
* status: Filter by status (active, inactive)
* sortBy: Field to sort by
* sortOrder: Sort order (asc, desc)

**Response:**

{

"success": true,

"data": {

"programs": [

{

"id": "60d21b4667d0d8992e610d01",

"name": "Computer Science",

"level": "bachelor",

"universityId": "60d21b4667d0d8992e610c70",

"universityName": "University of Cambridge",

"country": "United Kingdom",

"duration": "3 years",

"tuitionFee": 25000,

"feeCurrency": "GBP",

"status": "active",

"createdAt": "2024-11-15T10:00:00Z"

},

// Additional programs...

],

"pagination": {

"page": 1,

"limit": 10,

"totalPages": 15,

"totalItems": 145

}

}

}

**GET /programs/:id**

Retrieves a specific program with detailed information.

**Response:**

{

"success": true,

"data": {

"program": {

"id": "60d21b4667d0d8992e610d01",

"name": "Computer Science",

"level": "bachelor",

"universityId": "60d21b4667d0d8992e610c70",

"university": {

"name": "University of Cambridge",

"shortName": "Cambridge",

"country": "United Kingdom",

"city": "Cambridge",

"logoUrl": "https://storage.intered.com/logos/cambridge.png"

},

"description": "The Computer Science course at Cambridge provides a mixture of theoretical knowledge and practical experience...",

"duration": "3 years",

"tuitionFee": 25000,

"feeCurrency": "GBP",

"requirements": {

"academicRequirements": "A-Levels with A\*A\*A including Mathematics",

"englishRequirements": {

"ielts": 7.0,

"toefl": 100

},

"additionalRequirements": "STEP Mathematics recommended"

},

"applicationDeadline": "2025-10-15",

"applicationFee": 75,

"applicationFeeCurrency": "GBP",

"startDates": ["2025-10-01"],

"careers": [

"Software Developer",

"Systems Analyst",

"Data Scientist",

"AI Researcher"

],

"admissionRate": 15,

"curriculumHighlights": [

"Algorithms",

"Artificial Intelligence",

"Computer Systems",

"Software Engineering"

],

"status": "active",

"createdAt": "2024-11-15T10:00:00Z",

"updatedAt": "2025-02-10T11:30:00Z"

}

}

}

**POST /programs**

Creates a new program.

**Request:**

{

"name": "Artificial Intelligence",

"level": "master",

"universityId": "60d21b4667d0d8992e610c70",

"description": "This Master's program focuses on the development of AI systems and their applications...",

"duration": "1 year",

"tuitionFee": 30000,

"feeCurrency": "GBP",

"requirements": {

"academicRequirements": "2:1 honours degree in Computer Science or related field",

"englishRequirements": {

"ielts": 7.0,

"toefl": 100

},

"additionalRequirements": "Strong mathematics background"

},

"applicationDeadline": "2025-07-31",

"applicationFee": 75,

"applicationFeeCurrency": "GBP",

"startDates": ["2025-10-01"],

"careers": [

"AI Engineer",

"Machine Learning Developer",

"Research Scientist",

"Data Scientist"

],

"curriculumHighlights": [

"Machine Learning",

"Natural Language Processing",

"Computer Vision",

"Reinforcement Learning"

]

}

**Response:**

{

"success": true,

"data": {

"program": {

"id": "60d21b4667d0d8992e610d02",

"name": "Artificial Intelligence",

"level": "master",

"universityId": "60d21b4667d0d8992e610c70",

"universityName": "University of Cambridge",

"createdAt": "2025-05-01T15:00:00Z"

}

}

}

**PUT /programs/:id**

Updates a program.

**Request:**

{

"description": "Updated description of the AI program...",

"tuitionFee": 32000,

"requirements": {

"academicRequirements": "2:1 honours degree in Computer Science, Mathematics, or related field",

"englishRequirements": {

"ielts": 7.0,

"toefl": 100

}

},

"applicationDeadline": "2025-08-15"

}

**Response:**

{

"success": true,

"data": {

"program": {

"id": "60d21b4667d0d8992e610d02",

"updatedAt": "2025-05-01T15:30:00Z"

}

}

}

**DELETE /programs/:id**

Deactivates a program (soft delete).

**Response:**

{

"success": true,

"message": "Program deactivated successfully"

}

**Application Management**

**GET /applications**

Retrieves all applications.

**Query Parameters:**

* page (default: 1): Page number
* limit (default: 10): Items per page
* search: Search by reference number or student name
* status: Filter by status (draft, submitted, under\_review, etc.)
* studentId: Filter by student
* agentId: Filter by agent
* universityId: Filter by university
* programId: Filter by program
* startDate: Filter by submission date (start)
* endDate: Filter by submission date (end)
* sortBy: Field to sort by
* sortOrder: Sort order (asc, desc)

**Response:**

{

"success": true,

"data": {

"applications": [

{

"id": "60d21b4667d0d8992e610d10",

"referenceNumber": "APP-2025-001",

"studentId": "60d21b4667d0d8992e610c86",

"studentName": "John Doe",

"universityId": "60d21b4667d0d8992e610c70",

"universityName": "University of Cambridge",

"programId": "60d21b4667d0d8992e610d01",

"programName": "Computer Science",

"status": "submitted",

"submittedAt": "2025-04-15T14:30:00Z",

"agentId": "60d21b4667d0d8992e610c95",

"agentName": "Global Education Agency",

"createdAt": "2025-04-10T09:00:00Z"

},

// Additional applications...

],

"pagination": {

"page": 1,

"limit": 10,

"totalPages": 6,

"totalItems": 58

}

}

}

**GET /applications/:id**

Retrieves a specific application with detailed information.

**Response:**

{

"success": true,

"data": {

"application": {

"id": "60d21b4667d0d8992e610d10",

"referenceNumber": "APP-2025-001",

"student": {

"id": "60d21b4667d0d8992e610c86",

"firstName": "John",

"lastName": "Doe",

"email": "john.doe@example.com",

"phone": "+1234567890"

},

"university": {

"id": "60d21b4667d0d8992e610c70",

"name": "University of Cambridge",

"country": "United Kingdom",

"logoUrl": "https://storage.intered.com/logos/cambridge.png"

},

"program": {

"id": "60d21b4667d0d8992e610d01",

"name": "Computer Science",

"level": "bachelor",

"duration": "3 years"

},

"agent": {

"id": "60d21b4667d0d8992e610c95",

"name": "Global Education Agency",

"email": "jane.smith@globaledu.com"

},

"status": "submitted",

"statusHistory": [

{

"status": "draft",

"date": "2025-04-10T09:00:00Z",

"updatedBy": "60d21b4667d0d8992e610c86"

},

{

"status": "submitted",

"date": "2025-04-15T14:30:00Z",

"updatedBy": "60d21b4667d0d8992e610c95"

}

],

"applicationData": {

"personalDetails": {

"firstName": "John",

"lastName": "Doe",

"dateOfBirth": "1998-05-15",

"nationality": "United States",

"passportNumber": "US123456789",

"address": {

"line1": "123 Main St",

"city": "Boston",

"state": "MA",

"postalCode": "02115",

"country": "United States"

}

},

"educationHistory": [

{

"institution": "Boston High School",

"qualification": "High School Diploma",

"fieldOfStudy": "General",

"startDate": "2014-09-01",

"endDate": "2018-06-30",

"grade": "A"

}

],

"testScores": [

{

"test": "IELTS",

"score": 7.5,

"dateObtained": "2024-11-15"

},

{

"test": "SAT",

"score": 1450,

"dateObtained": "2024-12-10"

}

],

"personalStatement": "I am applying to this program because...",

"references": [

{

"name": "Dr. Robert Brown",

"position": "Science Teacher",

"institution": "Boston High School",

"email": "r.brown@bostonhs.edu",

"phone": "+1234567888"

}

]

},

"documents": [

{

"id": "60d21b4667d0d8992e610d20",

"type": "transcript",

"name": "High School Transcript",

"status": "verified",

"uploadedAt": "2025-04-12T10:00:00Z",

"verifiedAt": "2025-04-16T09:30:00Z",

"verifiedBy": "60d21b4667d0d8992e610c85"

},

{

"id": "60d21b4667d0d8992e610d21",

"type": "passport",

"name": "Passport",

"status": "verified",

"uploadedAt": "2025-04-12T10:15:00Z",

"verifiedAt": "2025-04-16T09:35:00Z",

"verifiedBy": "60d21b4667d0d8992e610c85"

},

{

"id": "60d21b4667d0d8992e610d22",

"type": "ielts",

"name": "IELTS Certificate",

"status": "verified",

"uploadedAt": "2025-04-12T10:30:00Z",

"verifiedAt": "2025-04-16T09:40:00Z",

"verifiedBy": "60d21b4667d0d8992e610c85"

}

],

"applicationFee": {

"amount": 75,

"currency": "GBP",

"status": "paid",

"paidAt": "2025-04-14T15:30:00Z",

"paymentMethod": "credit\_card",

"transactionId": "pay\_12345678"

},

"offers": [],

"notes": [

{

"content": "Applicant has excellent academic background.",

"createdBy": "60d21b4667d0d8992e610c95",

"createdAt": "2025-04-15T14:00:00Z",

"isPrivate": true

}

],

"submittedAt": "2025-04-15T14:30:00Z",

"createdAt": "2025-04-10T09:00:00Z",

"updatedAt": "2025-04-15T14:30:00Z"

}

}

}

**POST /applications**

Creates a new application.

**Request:**

{

"studentId": "60d21b4667d0d8992e610c87",

"universityId": "60d21b4667d0d8992e610c70",

"programId": "60d21b4667d0d8992e610d01",

"agentId": "60d21b4667d0d8992e610c95",

"applicationData": {

"personalDetails": {

"firstName": "Jane",

"lastName": "Smith",

"dateOfBirth": "1999-03-20",

"nationality": "United Kingdom",

"passportNumber": "UK987654321",

"address": {

"line1": "456 High St",

"city": "London",

"postalCode": "E1 6AN",

"country": "United Kingdom"

}

},

"personalStatement": "I am applying to this program because I have a strong interest in computer science...",

"references": [

{

"name": "Prof. Sarah Williams",

"position": "Mathematics Teacher",

"institution": "London Academy",

"email": "s.williams@londonacademy.ac.uk",

"phone": "+4412345678"

}

]

},

"documentIds": ["60d21b4667d0d8992e610d23", "60d21b4667d0d8992e610d24"]

}

**Response:**

{

"success": true,

"data": {

"application": {

"id": "60d21b4667d0d8992e610d11",

"referenceNumber": "APP-2025-002",

"status": "draft",

"createdAt": "2025-05-01T16:00:00Z"

}

}

}

**PUT /applications/:id**

Updates an application.

**Request:**

{

"applicationData": {

"personalStatement": "Updated personal statement...",

"references": [

{

"name": "Prof. Sarah Williams",

"position": "Mathematics Teacher",

"institution": "London Academy",

"email": "s.williams@londonacademy.ac.uk",

"phone": "+4412345678"

},

{

"name": "Dr. James Johnson",

"position": "Computer Science Professor",

"institution": "London Tech Institute",

"email": "j.johnson@lti.ac.uk",

"phone": "+4487654321"

}

]

},

"documentIds": ["60d21b4667d0d8992e610d23", "60d21b4667d0d8992e610d24", "60d21b4667d0d8992e610d25"]

}

**Response:**

{

"success": true,

"data": {

"application": {

"id": "60d21b4667d0d8992e610d11",

"updatedAt": "2025-05-01T16:30:00Z"

}

}

}

**PUT /applications/:id/status**

Updates an application status.

**Request:**

{

"status": "submitted",

"notes": "Application submitted by student"

}

**Response:**

{

"success": true,

"data": {

"application": {

"id": "60d21b4667d0d8992e610d11",

"status": "submitted",

"submittedAt": "2025-05-01T17:00:00Z",

"updatedAt": "2025-05-01T17:00:00Z"

}

}

}

**POST /applications/:id/notes**

Adds a note to an application.

**Request:**

{

"content": "Student's academic credentials are impressive.",

"isPrivate": true

}

**Response:**

{

"success": true,

"data": {

"note": {

"id": "60d21b4667d0d8992e610d30",

"content": "Student's academic credentials are impressive.",

"createdBy": "60d21b4667d0d8992e610c85",

"createdAt": "2025-05-01T17:30:00Z",

"isPrivate": true

}

}

}

**POST /applications/:id/offers**

Adds an offer to an application.

**Request:**

{

"type": "conditional",

"offerDate": "2025-05-10T09:00:00Z",

"responseDeadline": "2025-06-10T23:59:59Z",

"conditions": [

{

"description": "Achieve a minimum of AAB in A-Levels including Mathematics at grade A"

},

{

"description": "Provide passport photocopy"

}

],

"documentId": "60d21b4667d0d8992e610d40"

}

**Response:**

{

"success": true,

"data": {

"offer": {

"id": "60d21b4667d0d8992e610d50",

"type": "conditional",

"offerDate": "2025-05-10T09:00:00Z",

"responseDeadline": "2025-06-10T23:59:59Z",

"createdAt": "2025-05-01T18:00:00Z"

}

}

}

**PUT /applications/:id/offers/:offerId/response**

Updates a student's response to an offer.

**Request:**

{

"status": "accepted",

"date": "2025-05-15T14:00:00Z",

"notes": "Student has accepted the offer"

}

**Response:**

{

"success": true,

"data": {

"offer": {

"id": "60d21b4667d0d8992e610d50",

"studentResponse": {

"status": "accepted",

"date": "2025-05-15T14:00:00Z"

},

"updatedAt": "2025-05-01T18:30:00Z"

}

}

}

**Document Management**

**GET /documents**

Retrieves all documents.

**Query Parameters:**

* page (default: 1): Page number
* limit (default: 10): Items per page
* ownerId: Filter by owner
* ownerType: Filter by owner type (student, agent, university)
* type: Filter by document type
* status: Filter by verification status (pending, verified, rejected)
* sortBy: Field to sort by
* sortOrder: Sort order (asc, desc)

**Response:**

{

"success": true,

"data": {

"documents": [

{

"id": "60d21b4667d0d8992e610d20",

"ownerId": "60d21b4667d0d8992e610c86",

"ownerType": "student",

"ownerName": "John Doe",

"type": "transcript",

"name": "High School Transcript",

"fileType": "application/pdf",

"fileSize": 1234567,

"status": "verified",

"uploadedAt": "2025-04-12T10:00:00Z",

"verifiedAt": "2025-04-16T09:30:00Z"

},

// Additional documents...

],

"pagination": {

"page": 1,

"limit": 10,

"totalPages": 10,

"totalItems": 95

}

}

}

**GET /documents/:id**

Retrieves a specific document with detailed information.

**Response:**

{

"success": true,

"data": {

"document": {

"id": "60d21b4667d0d8992e610d20",

"ownerId": "60d21b4667d0d8992e610c86",

"ownerType": "student",

"owner": {

"id": "60d21b4667d0d8992e610c86",

"firstName": "John",

"lastName": "Doe",

"email": "john.doe@example.com"

},

"relatedId": "60d21b4667d0d8992e610d10",

"relatedType": "application",

"type": "transcript",

"name": "High School Transcript",

"filePath": "students/60d21b4667d0d8992e610c86/transcript.pdf",

"fileUrl": "https://storage.intered.com/documents/students/60d21b4667d0d8992e610c86/transcript.pdf",

"fileType": "application/pdf",

"fileSize": 1234567,

"status": "verified",

"statusHistory": [

{

"status": "pending",

"date": "2025-04-12T10:00:00Z",

"updatedBy": "60d21b4667d0d8992e610c86"

},

{

"status": "verified",

"date": "2025-04-16T09:30:00Z",

"updatedBy": "60d21b4667d0d8992e610c85",

"comments": "Document verified successfully"

}

],

"verifiedBy": {

"id": "60d21b4667d0d8992e610c85",

"firstName": "Admin",

"lastName": "User"

},

"verifiedAt": "2025-04-16T09:30:00Z",

"metadata": {

"issueDate": "2018-06-30",

"issuedBy": "Boston High School"

},

"uploadedBy": "60d21b4667d0d8992e610c86",

"uploadedAt": "2025-04-12T10:00:00Z",

"updatedAt": "2025-04-16T09:30:00Z"

}

}

}

**POST /documents**

Uploads a new document.

**Request:** Form data with the following fields:

* file: File to upload
* ownerId: Document owner ID
* ownerType: Document owner type (student, agent, university)
* relatedId (optional): Related entity ID (e.g., application)
* relatedType (optional): Related entity type (e.g., application)
* type: Document type
* name: Document name
* metadata (optional): Additional metadata as JSON string

**Response:**

{

"success": true,

"data": {

"document": {

"id": "60d21b4667d0d8992e610d26",

"ownerId": "60d21b4667d0d8992e610c87",

"ownerType": "student",

"type": "passport",

"name": "Passport",

"fileUrl": "https://storage.intered.com/documents/students/60d21b4667d0d8992e610c87/passport.pdf",

"status": "pending",

"uploadedAt": "2025-05-01T19:00:00Z"

}

}

}

**PUT /documents/:id/verify**

Verifies a document.

**Request:**

{

"status": "verified",

"comments": "Document verified successfully"

}

**Response:**

{

"success": true,

"data": {

"document": {

"id": "60d21b4667d0d8992e610d26",

"status": "verified",

"verifiedBy": "60d21b4667d0d8992e610c85",

"verifiedAt": "2025-05-01T19:30:00Z",

"updatedAt": "2025-05-01T19:30:00Z"

}

}

}

**DELETE /documents/:id**

Deletes a document.

**Response:**

{

"success": true,

"message": "Document deleted successfully"

}

**Commission Management**

**GET /commissions**

Retrieves all commissions.

**Query Parameters:**

* page (default: 1): Page number
* limit (default: 10): Items per page
* agentId: Filter by agent
* status: Filter by status (pending, approved, paid, rejected)
* startDate: Filter by date (start)
* endDate: Filter by date (end)
* sortBy: Field to sort by
* sortOrder: Sort order (asc, desc)

**Response:**

{

"success": true,

"data": {

"commissions": [

{

"id": "60d21b4667d0d8992e610e01",

"agentId": "60d21b4667d0d8992e610c95",

"agentName": "Global Education Agency",

"applicationId": "60d21b4667d0d8992e610d10",

"studentName": "John Doe",

"universityName": "University of Cambridge",

"programName": "Computer Science",

"amount": 1250,

"currency": "GBP",

"status": "approved",

"createdAt": "2025-04-20T10:00:00Z"

},

// Additional commissions...

],

"pagination": {

"page": 1,

"limit": 10,

"totalPages": 3,

"totalItems": 28

}

}

}

**GET /commissions/:id**

Retrieves a specific commission with detailed information.

**Response:**

{

"success": true,

"data": {

"commission": {

"id": "60d21b4667d0d8992e610e01",

"agent": {

"id": "60d21b4667d0d8992e610c95",

"name": "Global Education Agency",

"email": "jane.smith@globaledu.com"

},

"application": {

"id": "60d21b4667d0d8992e610d10",

"referenceNumber": "APP-2025-001"

},

"student": {

"id": "60d21b4667d0d8992e610c86",

"firstName": "John",

"lastName": "Doe"

},

"university": {

"id": "60d21b4667d0d8992e610c70",

"name": "University of Cambridge"

},

"program": {

"id": "60d21b4667d0d8992e610d01",

"name": "Computer Science",

"level": "bachelor"

},

"amount": 1250,

"currency": "GBP",

"rate": 5,

"status": "approved",

"statusHistory": [

{

"status": "pending",

"date": "2025-04-20T10:00:00Z"

},

{

"status": "approved",

"date": "2025-04-25T14:30:00Z",

"updatedBy": "60d21b4667d0d8992e610c85"

}

],

"approvalDetails": {

"appro# InterEd Admin Portal API Specification

## Overview

This document provides a comprehensive specification for the InterEd Student Recruitment Admin Portal API. It defines all endpoints, request/response formats, authentication mechanisms, and error handling patterns necessary for implementing the backend services for the admin portal.

The API is designed following RESTful principles with JWT authentication and is intended to be implemented using Express.js and integrated with Supabase.

## Base URL

https://api.intered.com/v1

For development environments:

http://localhost:5000/api/v1

## Authentication

### Authentication Mechanism

The API uses JSON Web Tokens (JWT) for authentication:

1. All authenticated requests must include an `Authorization` header with a Bearer token

2. Tokens expire after 1 hour

3. Refresh tokens are used for obtaining new access tokens

4. Role-based permissions control access to resources

### Authentication Endpoints

#### POST /auth/login

Authenticates a user and returns a JWT token.

\*\*Request:\*\*

```json

{

"email": "admin@intered.com",

"password": "securepassword"

}

**Response:**

{

"success": true,

"data": {

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",

"refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",

"expiresIn": 3600,

"user": {

"id": "60d21b4667d0d8992e610c85",

"email": "admin@intered.com",

"firstName": "Admin",

"lastName": "User",

"role": "admin"

}

}

}

**POST /auth/refresh**

Refreshes an expired JWT token.

**Request:**

{

"refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."

}

**Response:**

{

"success": true,

"data": {

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",

"refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",

"expiresIn": 3600

}

}

**POST /auth/register**

Registers a new user (admin-only operation).

**Request:**

{

"email": "staff@intered.com",

"password": "securepassword",

"firstName": "Staff",

"lastName": "User",

"role": "staff"

}

**Response:**

{

"success": true,

"data": {

"user": {

"id": "60d21b4667d0d8992e610c85",

"email": "staff@intered.com",

"firstName": "Staff",

"lastName": "User",

"role": "staff"

}

}

}

**POST /auth/forgot-password**

Initiates password reset process.

**Request:**

{

"email": "admin@intered.com"

}

**Response:**

{

"success": true,

"message": "Password reset instructions sent to email"

}

**POST /auth/reset-password**

Resets password using reset token.

**Request:**

{

"token": "reset-token-from-email",

"newPassword": "newsecurepassword"

}

**Response:**

{

"success": true,

"message": "Password reset successfully"

}

**POST /auth/logout**

Invalidates the current JWT token.

**Response:**

{

"success": true,

"message": "Logged out successfully"

}

**Core API Endpoints**

**Users and Profile Management**

**GET /users**

Retrieves all users (admin-only).

**Query Parameters:**

* page (default: 1): Page number
* limit (default: 10): Items per page
* role: Filter by role (admin, staff, agent, student)
* search: Search by name or email
* sortBy: Field to sort by
* sortOrder: Sort order (asc, desc)

**Response:**

{

"success": true,

"data": {

"users": [

{

"id": "60d21b4667d0d8992e610c85",

"email": "admin@intered.com",

"firstName": "Admin",

"lastName": "User",

"role": "admin",

"lastLogin": "2025-05-01T10:30:00Z",

"status": "active",

"createdAt": "2025-01-01T10:00:00Z"

},

// Additional users...

],

"pagination": {

"page": 1,

"limit": 10,

"totalPages": 5,

"totalItems": 45

}

}

}

**GET /users/:id**

Retrieves a specific user.

**Response:**

{

"success": true,

"data": {

"user": {

"id": "60d21b4667d0d8992e610c85",

"email": "admin@intered.com",

"firstName": "Admin",

"lastName": "User",

"role": "admin",

"phone": "+1234567890",

"avatar": "https://storage.intered.com/avatars/admin.jpg",

"lastLogin": "2025-05-01T10:30:00Z",

"status": "active",

"createdAt": "2025-01-01T10:00:00Z",

"updatedAt": "2025-05-01T10:30:00Z"

}

}

}

**PUT /users/:id**

Updates a user.

**Request:**

{

"firstName": "Updated",

"lastName": "Name",

"phone": "+1234567890",

"status": "active"

}

**Response:**

{

"success": true,

"data": {

"user": {

"id": "60d21b4667d0d8992e610c85",

"firstName": "Updated",

"lastName": "Name",

"phone": "+1234567890",

"status": "active",

"updatedAt": "2025-05-01T10:35:00Z"

}

}

}

**DELETE /users/:id**

Deactivates a user (soft delete).

**Response:**

{

"success": true,

"message": "User deactivated successfully"

}

**POST /users/:id/avatar**

Uploads a user avatar.

**Request:** Form data with file field avatar

**Response:**

{

"success": true,

"data": {

"avatarUrl": "https://storage.intered.com/avatars/admin.jpg"

}

}

**Student Management**

**GET /students**

Retrieves all students.

**Query Parameters:**

* page (default: 1): Page number
* limit (default: 10): Items per page
* search: Search by name or email
* status: Filter by status (active, inactive, graduated, withdrawn)
* agentId: Filter by agent
* sortBy: Field to sort by
* sortOrder: Sort order (asc, desc)

**Response:**

{

"success": true,

"data": {

"students": [

{

"id": "60d21b4667d0d8992e610c86",

"profileId": "60d21b4667d0d8992e610c90",

"firstName": "John",

"lastName": "Doe",

"email": "john.doe@example.com",

"phone": "+1234567890",

"agentId": "60d21b4667d0d8992e610c95",

"agentName": "Global Education Agency",

"status": "active",

"applicationCount": 2,

"createdAt": "2025-01-15T10:00:00Z"

},

// Additional students...

],

"pagination": {

"page": 1,

"limit": 10,

"totalPages": 8,

"totalItems": 76

}

}

}

**GET /students/:id**

Retrieves a specific student with detailed information.

**Response:**

{

"success": true,

"data": {

"student": {

"id": "60d21b4667d0d8992e610c86",

"profileId": "60d21b4667d0d8992e610c90",

"firstName": "John",

"lastName": "Doe",

"email": "john.doe@example.com",

"phone": "+1234567890",

"dateOfBirth": "1998-05-15",

"nationality": "United States",

"address": {

"line1": "123 Main St",

"line2": "Apt 4B",

"city": "Boston",

"state": "MA",

"postalCode": "02115",

"country": "United States"

},

"academicInfo": {

"highestEducation": "High School",

"gpa": 3.8,

"institutions": [

{

"name": "Boston High School",

"degree": "High School Diploma",

"fieldOfStudy": "General",

"startDate": "2014-09-01",

"endDate": "2018-06-30",

"grade": "A"

}

]

},

"testScores": {

"ielts": {

"overall": 7.5,

"listening": 7.5,

"reading": 8.0,

"writing": 7.0,

"speaking": 7.5,

"testDate": "2024-11-15"

}

},

"preferences": {

"destinations": ["UK", "Canada", "Australia"],

"programLevels": ["Undergraduate"],

"fieldOfStudy": ["Computer Science", "Information Technology"],

"budget": {

"amount": 30000,

"currency": "USD"

}

},

"agent": {

"id": "60d21b4667d0d8992e610c95",

"name": "Global Education Agency",

"counselorName": "Jane Smith",

"email": "jane.smith@globaledu.com"

},

"status": "active",

"createdAt": "2025-01-15T10:00:00Z",

"updatedAt": "2025-04-20T14:30:00Z"

}

}

}

**POST /students**

Creates a new student.

**Request:**

{

"firstName": "Jane",

"lastName": "Smith",

"email": "jane.smith@example.com",

"phone": "+1987654321",

"dateOfBirth": "1999-03-20",

"nationality": "United Kingdom",

"address": {

"line1": "456 High St",

"city": "London",

"postalCode": "E1 6AN",

"country": "United Kingdom"

},

"academicInfo": {

"highestEducation": "High School",

"gpa": 4.0,

"institutions": [

{

"name": "London Academy",

"degree": "A-Levels",

"fieldOfStudy": "Sciences",

"startDate": "2016-09-01",

"endDate": "2018-06-30",

"grade": "A\*"

}

]

},

"agentId": "60d21b4667d0d8992e610c95"

}

**Response:**

{

"success": true,

"data": {

"student": {

"id": "60d21b4667d0d8992e610c87",

"email": "jane.smith@example.com",

"firstName": "Jane",

"lastName": "Smith",

"createdAt": "2025-05-01T11:00:00Z"

}

}

}

**PUT /students/:id**

Updates a student.

**Request:**

{

"phone": "+1987654322",

"address": {

"line1": "789 New Ave",

"city": "London",

"postalCode": "E2 8LP",

"country": "United Kingdom"

},

"preferences": {

"destinations": ["UK", "Canada"],

"programLevels": ["Undergraduate"],

"fieldOfStudy": ["Computer Science"]

}

}

**Response:**

{

"success": true,

"data": {

"student": {

"id": "60d21b4667d0d8992e610c87",

"updatedAt": "2025-05-01T11:30:00Z"

}

}

}

**DELETE /students/:id**

Deactivates a student (soft delete).

**Response:**

{

"success": true,

"message": "Student deactivated successfully"

}

**Agent Management**

**GET /agents**

Retrieves all agents.

**Query Parameters:**

* page (default: 1): Page number
* limit (default: 10): Items per page
* search: Search by name, agency name, or email
* status: Filter by status (active, inactive, suspended)
* verificationStatus: Filter by verification status (pending, verified, rejected)
* sortBy: Field to sort by
* sortOrder: Sort order (asc, desc)

**Response:**

{

"success": true,

"data": {

"agents": [

{

"id": "60d21b4667d0d8992e610c95",

"profileId": "60d21b4667d0d8992e610c96",

"firstName": "Jane",

"lastName": "Smith",

"email": "jane.smith@globaledu.com",

"phone": "+1234567890",

"agencyName": "Global Education Agency",

"agencyType": "company",

"verificationStatus": "verified",

"status": "active",

"studentCount": 15,

"applicationCount": 28,

"createdAt": "2024-12-01T10:00:00Z"

},

// Additional agents...

],

"pagination": {

"page": 1,

"limit": 10,

"totalPages": 3,

"totalItems": 25

}

}

}

**GET /agents/:id**

Retrieves a specific agent with detailed information.

**Response:**

{

"success": true,

"data": {

"agent": {

"id": "60d21b4667d0d8992e610c95",

"profileId": "60d21b4667d0d8992e610c96",

"firstName": "Jane",

"lastName": "Smith",

"email": "jane.smith@globaledu.com",

"phone": "+1234567890",

"agencyName": "Global Education Agency",

"agencyType": "company",

"businessInfo": {

"registrationNumber": "GEA12345",

"foundingYear": 2015,

"website": "https://www.globaledu.com",

"address": {

"line1": "100 Education St",

"city": "New York",

"state": "NY",

"postalCode": "10001",

"country": "United States"

}

},

"specializations": {

"countries": ["UK", "USA", "Canada", "Australia"],

"programLevels": ["Undergraduate", "Postgraduate"],

"disciplines": ["Business", "Engineering", "Computer Science"]

},

"verificationStatus": "verified",

"verificationDate": "2025-01-10T09:30:00Z",

"verifiedBy": "60d21b4667d0d8992e610c85",

"status": "active",

"commissionTier": "gold",

"parentAgentId": null,

"subAgents": [

{

"id": "60d21b4667d0d8992e610c97",

"name": "John Johnson",

"email": "john.johnson@globaledu.com",

"studentCount": 8

}

],

"performance": {

"studentCount": 15,

"applicationCount": 28,

"conversionRate": 85,

"commissionEarned": {

"amount": 25000,

"currency": "USD"

}

},

"agreementDetails": {

"startDate": "2025-01-01",

"endDate": "2025-12-31",

"documentId": "60d21b4667d0d8992e610c98"

},

"createdAt": "2024-12-01T10:00:00Z",

"updatedAt": "2025-04-15T15:30:00Z"

}

}

}

**POST /agents**

Creates a new agent.

**Request:**

{

"firstName": "Robert",

"lastName": "Johnson",

"email": "robert.johnson@studyabroad.com",

"phone": "+1122334455",

"agencyName": "Study Abroad Experts",

"agencyType": "company",

"businessInfo": {

"registrationNumber": "SAE54321",

"foundingYear": 2018,

"website": "https://www.studyabroad.com",

"address": {

"line1": "200 Education Blvd",

"city": "Chicago",

"state": "IL",

"postalCode": "60601",

"country": "United States"

}

},

"specializations": {

"countries": ["USA", "Canada"],

"programLevels": ["Undergraduate", "Postgraduate"],

"disciplines": ["Business", "Arts"]

}

}

**Response:**

{

"success": true,

"data": {

"agent": {

"id": "60d21b4667d0d8992e610c99",

"email": "robert.johnson@studyabroad.com",

"firstName": "Robert",

"lastName": "Johnson",

"agencyName": "Study Abroad Experts",

"verificationStatus": "pending",

"createdAt": "2025-05-01T12:30:00Z"

}

}

}

**PUT /agents/:id**

Updates an agent.

**Request:**

{

"phone": "+1122334466",

"businessInfo": {

"website": "https://www.studyabroadexperts.com",

"address": {

"line1": "300 Education Ctr",

"city": "Chicago",

"state": "IL",

"postalCode": "60602",

"country": "United States"

}

},

"specializations": {

"countries": ["USA", "Canada", "UK"],

"programLevels": ["Undergraduate", "Postgraduate", "Diploma"],

"disciplines": ["Business", "Arts", "Technology"]

}

}

**Response:**

{

"success": true,

"data": {

"agent": {

"id": "60d21b4667d0d8992e610c99",

"updatedAt": "2025-05-01T13:00:00Z"

}

}

}

**PUT /agents/:id/verify**

Verifies an agent (admin/staff only).

**Request:**

{

"verificationStatus": "verified",

"commissionTier": "silver",

"notes": "Documents verified successfully"

}

**Response:**

{

"success": true,

"data": {

"agent": {

"id": "60d21b4667d0d8992e610c99",

"verificationStatus": "verified",

"verificationDate": "2025-05-01T13:30:00Z",

"verifiedBy": "60d21b4667d0d8992e610c85",

"commissionTier": "silver"

}

}

}

**DELETE /agents/:id**

Deactivates an agent (soft delete).

**Response:**

{

"success": true,

"message": "Agent deactivated successfully"

}

**University Management**

**GET /universities**

Retrieves all universities.

**Query Parameters:**

* page (default: 1): Page number
* limit (default: 10): Items per page
* search: Search by name or country
* country: Filter by country
* status: Filter by status (active, inactive)
* sortBy: Field to sort by
* sortOrder: Sort order (asc, desc)

**Response:**

{

"success": true,

"data": {

"universities": [

{

"id": "60d21b4667d0d8992e610c70",

"name": "University of Cambridge",

"shortName": "Cambridge",

"country": "United Kingdom",

"city": "Cambridge",

"ranking": 5,

"programsCount": 150,

"logoUrl": "https://storage.intered.com/logos/cambridge.png",

"status": "active",

"createdAt": "2024-10-01T10:00:00Z"

},

// Additional universities...

],

"pagination": {

"page": 1,

"limit": 10,

"totalPages": 5,

"totalItems": 48

}

}

}

**GET /universities/:id**

Retrieves a specific university with detailed information.

**Response:**

{

"success": true,

"data": {

"university": {

"id": "60d21b4667d0d8992e610c70",

"name": "University of Cambridge",

"shortName": "Cambridge",

"country": "United Kingdom",

"city": "Cambridge",

"ranking": {

"world": 5,

"national": 2

},

"logoUrl": "https://storage.intered.com/logos/cambridge.png",

"website": "https://www.cam.ac.uk",

"description": "The University of Cambridge is a collegiate research university in Cambridge, United Kingdom. Founded in 1209, it is the world's third-oldest university in continuous operation.",

"admissionRequirements": {

"undergraduate": {

"academicRequirements": "A-Levels with AAA or equivalent",

"englishRequirements": {

"ielts": 7.0,

"toefl": 100

},

"additionalRequirements": "Admissions test for most courses"

},

"postgraduate": {

"academicRequirements": "2:1 honours degree or equivalent",

"englishRequirements": {

"ielts": 7.5,

"toefl": 110

},

"additionalRequirements": "Research proposal for research degrees"

}

},

"tuitionFees": {

"undergraduate": {

"domestic": 9250,

"international": 25000

},

"postgraduate": {

"domestic": 12000,

"international": 30000

},

"currency": "GBP"

},

"programsCount": 150,

"applicationDeadlines": {

"undergraduate": "2025-10-15",

"postgraduate": "2025-07-31"

},

"contacts": [

{

"name": "Admissions Office",

"email": "admissions@cam.ac.uk",

"phone": "+44123456789"

}

],

"agreementDetails": {

"startDate": "2025-01-01",

"endDate": "2027-12-31",

"commissionRates": {

"undergraduate": 5,

"postgraduate": 7

},

"documentId": "60d21b4667d0d8992e610c71"

},

"status": "active",

"createdAt": "2024-10-01T10:00:00Z",

"updatedAt": "2025-03-15T14:30:00Z"

}

}

}

**POST /universities**

Creates a new university.

**Request:**

{

"name": "Imperial College London",

"shortName": "Imperial",

"country": "United Kingdom",

"city": "London",

"ranking": {

"world": 8,

"national": 3

},

"website": "https://www.imperial.ac.uk",

"description": "Imperial College London is a public research university in London focusing on science, engineering, medicine, and business.",

"admissionRequirements": {

"undergraduate": {

"academicRequirements": "A-Levels with A\*AA or equivalent",

"englishRequirements": {

"ielts": 6.5,

"toefl": 92

}

},

"postgraduate": {

"academicRequirements": "2:1 honours degree or equivalent",

"englishRequirements": {

"ielts": 7.0,

"toefl": 100

}

}

},

"tuitionFees": {

"undergraduate": {

"domestic": 9250,

"international": 29000

},

"postgraduate": {

"domestic": 14000,

"international": 32000

},

"currency": "GBP"

},

"applicationDeadlines": {

"undergraduate": "2025-10-15",

"postgraduate": "2025-06-30"

},

"contacts": [

{

"name": "Admissions Office",

"email": "admissions@imperial.ac.uk",

"phone": "+44987654321"

}

]

}

**Response:**

{

"success": true,

"data": {

**InterEd Admin Portal Team Structure & Responsibilities**

**Overview**

This document outlines the recommended team structure, roles, responsibilities, and collaboration workflow for the 48-hour InterEd Admin Portal Vibathon. A well-defined team structure with clear responsibilities is critical for maximizing productivity during the short development timeframe.

**Team Size and Composition**

For optimal performance during the 48-hour Vibathon, we recommend a team of **6-8 members** with the following composition:

**Core Team Roles**

1. **Product Architect** (1 person)
2. **UI/UX Lead** (1 person)
3. **Frontend Developers** (2-3 people)
4. **Backend Specialists** (1-2 people)
5. **Full-Stack Developer** (1 person)
6. **QA Specialist** (1 person)

**Detailed Role Descriptions**

**1. Product Architect**

**Responsibilities:**

* Overall project vision and architecture
* Decision-making authority for technical tradeoffs
* Requirements clarification and prioritization
* Team coordination and task assignment
* Ensuring alignment with business objectives
* Managing project timeline and milestones
* Blockers removal and risk management

**Skills Required:**

* Strong understanding of the education recruitment domain
* Experience with MERN stack architecture
* Excellent communication and leadership skills
* Ability to make quick, informed decisions
* Previous hackathon or rapid development experience

**Pre-Vibathon Tasks:**

* Finalize technical architecture
* Set up project repository and structure
* Create detailed component breakdown
* Define API specifications
* Prepare development environment
* Configure CI/CD pipeline
* Establish coding standards and conventions

**2. UI/UX Lead**

**Responsibilities:**

* Design system implementation
* Component library creation
* User flow optimization
* Visual consistency across the portal
* Responsive design implementation
* Accessibility compliance
* Design-to-code collaboration

**Skills Required:**

* Proficiency with Tailwind CSS
* Experience with React component design
* Understanding of design systems
* Knowledge of UI/UX best practices
* Ability to translate requirements into intuitive interfaces

**Pre-Vibathon Tasks:**

* Extract design tokens from requirements
* Set up Tailwind configuration
* Create component mockups and prototypes
* Define responsive design approach
* Build core UI components
* Document design patterns and usage guidelines
* Prepare component library documentation

**3. Frontend Developers**

**Responsibilities:**

* Implementation of React components
* State management with Redux
* API integration
* Form validation
* Client-side business logic
* Performance optimization
* Cross-browser compatibility

**Skills Required:**

* Strong React.js experience
* Proficiency with Redux/Redux Toolkit
* Familiarity with React Router
* Experience with form libraries (Formik, React Hook Form)
* Knowledge of frontend testing frameworks
* Comfort with modern JavaScript (ES6+)

**Pre-Vibathon Tasks:**

* Set up frontend project structure
* Implement authentication flow
* Create reusable form components
* Build table and data display components
* Implement navigation and routing
* Set up Redux store and slices
* Create API service layer

**4. Backend Specialists**

**Responsibilities:**

* Supabase configuration and optimization
* Database schema implementation
* API endpoint development
* Authentication and authorization
* Data validation and sanitization
* File storage and management
* Performance optimization

**Skills Required:**

* Experience with Supabase or PostgreSQL
* Knowledge of RESTful API design
* Understanding of authentication mechanisms
* Familiarity with data modeling
* Security best practices awareness
* Performance optimization techniques

**Pre-Vibathon Tasks:**

* Set up Supabase project
* Implement database schema
* Create table relationships and constraints
* Configure row-level security policies
* Set up storage buckets with access controls
* Create initial API endpoints
* Implement authentication flow

**5. Full-Stack Developer**

**Responsibilities:**

* Cross-functional implementation tasks
* Integration between frontend and backend
* Feature implementation end-to-end
* Support for both frontend and backend teams
* Utility functions and helper services
* Documentation maintenance
* Testing support

**Skills Required:**

* Well-rounded knowledge of MERN stack
* Experience with Supabase
* Ability to work on both frontend and backend
* Problem-solving skills
* Adaptability to changing requirements
* Quick learning ability

**Pre-Vibathon Tasks:**

* Assist with both frontend and backend setup
* Implement shared utilities and helpers
* Create connection between frontend and backend
* Set up testing framework
* Prepare documentation templates
* Configure deployment pipeline
* Support other team members as needed

**6. QA Specialist**

**Responsibilities:**

* Testing strategy development
* Test case creation
* Manual testing execution
* Bug tracking and reporting
* Regression testing
* User acceptance testing
* Quality assurance

**Skills Required:**

* Experience with frontend and API testing
* Knowledge of testing methodologies
* Attention to detail
* Strong problem identification skills
* Good documentation practices
* Understanding of user expectations

**Pre-Vibathon Tasks:**

* Create test plan and test cases
* Set up testing environment
* Define acceptance criteria
* Prepare test data
* Configure bug tracking system
* Create testing documentation
* Develop testing checklists

**Team Collaboration Structure**

**Agile Framework**

For the 48-hour Vibathon, we recommend a streamlined Agile approach with:

* **4-hour sprints**: Short, focused work cycles
* **15-minute stand-ups**: Quick sync at the beginning of each sprint
* **Continuous integration**: Regular code merges to main branch
* **Feature-based workflow**: Complete features rather than components
* **Pair programming**: For complex features
* **Async communication**: Chat tools for non-urgent matters
* **War room**: Dedicated space for critical problem-solving

**Development Workflow**

1. **Feature Breakdown**:
   * Product Architect breaks down features into tasks
   * Tasks are sized and prioritized
   * Developers select tasks based on expertise
2. **Implementation Flow**:
   * UI/UX Lead creates component designs
   * Frontend Developers implement UI components
   * Backend Specialists create APIs
   * Full-Stack Developer integrates frontend and backend
   * QA Specialist tests completed features
3. **Code Review Process**:
   * Pull request for each completed task
   * Code reviewed by at least one team member
   * Automated tests must pass
   * Product Architect approves architectural changes
   * Merge to main branch upon approval
4. **Release Cycle**:
   * Continuous deployment to development environment
   * Milestone deployments to staging every 8 hours
   * Final production deployment at the end of Vibathon

**Shift Planning**

For a 48-hour continuous development cycle, team members need proper rest. We recommend:

**Shift Schedule**

**Day 1 (First 24 hours):**

* **All hands**: First 12 hours (setting up, architecture, core features)
* **Team A**: Hours 12-24 (Frontend Developers, Backend Specialist)
* **Team B**: Rest during hours 12-24

**Day 2 (Second 24 hours):**

* **Team B**: Hours 24-36 (UI/UX Lead, Full-Stack Developer, QA Specialist)
* **Team A**: Rest during hours 24-36
* **All hands**: Final 12 hours (integration, testing, deployment)

**Handover Process**

* 30-minute handover session between shifts
* Documented progress in shared workspace
* Recorded video explanations for complex features
* Clear documentation of current blockers
* Prioritized task list for incoming team

**Tool Stack and Access Management**

**Development Tools**

* **Version Control**: GitHub
* **Project Management**: GitHub Projects
* **Communication**: Slack and Discord
* **Design Tools**: Figma
* **Documentation**: Notion or GitHub Wiki
* **Code Editors**: Cursor, Replit, VSCode with GitHub Copilot
* **Backend**: Supabase
* **Deployment**: Vercel

**Access Requirements**

Ensure all team members have:

* GitHub repository access
* Supabase project access
* Vercel project access
* Slack/Discord channel invitations
* Figma design access
* Documentation access
* Environment credentials

**Phase-Based Team Allocation**

**Phase 1: Foundation (Hours 0-12)**

| **Focus Area** | **Team Members** | **Key Deliverables** |
| --- | --- | --- |
| Project Setup | Product Architect, Full-Stack Developer | Repository, CI/CD, environments |
| Database Implementation | Backend Specialists | Schema, tables, relationships |
| Core UI Components | UI/UX Lead, Frontend Developers | Design system, component library |
| Authentication | Backend Specialist, Frontend Developer | Login, registration, auth flow |

**Phase 2: Core Features (Hours 12-24)**

| **Focus Area** | **Team Members** | **Key Deliverables** |
| --- | --- | --- |
| Dashboard | Frontend Developer, Full-Stack Developer | Dashboard layout, metrics, charts |
| Student Management | Frontend Developer, Backend Specialist | Student listing, details, CRUD |
| User Management | Frontend Developer, Backend Specialist | User administration, permissions |
| Navigation | UI/UX Lead | Header, sidebar, breadcrumbs |

**Phase 3: Advanced Features (Hours 24-36)**

| **Focus Area** | **Team Members** | **Key Deliverables** |
| --- | --- | --- |
| Agent Management | Frontend Developer, Backend Specialist | Agent listing, details, verification |
| University Management | Frontend Developer, Backend Specialist | University/program management |
| Application Processing | Full-Stack Developer | Application workflow, status updates |
| Reporting | Frontend Developer, Backend Specialist | Analytics, reports, exports |

**Phase 4: Integration & Completion (Hours 36-48)**

| **Focus Area** | **Team Members** | **Key Deliverables** |
| --- | --- | --- |
| Testing | QA Specialist, All Developers | Bug fixes, regression testing |
| Documentation | Product Architect, UI/UX Lead | User guide, API docs, handover notes |
| Performance Optimization | Full-Stack Developer, Backend Specialist | Speed improvements, caching |
| Deployment | Product Architect, Backend Specialist | Production deployment, verification |

**Conflict Resolution Process**

Given the intense nature of a 48-hour hackathon, conflicts may arise. To address them efficiently:

1. **Technical Disagreements**: Product Architect has final decision-making authority
2. **Priority Conflicts**: Focus on critical path features first
3. **Resource Allocation**: Product Architect reallocates resources as needed
4. **Scope Challenges**: Reduce scope rather than quality when time constrained
5. **Team Dynamics**: Address interpersonal issues immediately via private conversation

**Communication Protocols**

**Regular Check-ins**

* **Sprint Planning**: Start of each 4-hour sprint
* **Stand-ups**: Every 2 hours (15 minutes maximum)
* **Blockers Session**: As needed, dedicated time to solve obstacles
* **End-of-day Review**: Assess progress and adjust plans

**Communication Channels**

* **Urgent Issues**: Voice call or in-person
* **Technical Questions**: Development channel
* **Status Updates**: Stand-up channel
* **Documentation**: Wiki or shared documents
* **Code-related**: GitHub issues and pull requests
* **General Discussion**: General channel

**Success Metrics**

The team should track these metrics to ensure productive Vibathon progress:

* **Feature Completion Rate**: Percentage of planned features completed
* **Bug Count**: Number of identified vs. resolved bugs
* **Build Status**: Green/red status of CI pipeline
* **Test Coverage**: Percentage of code covered by tests
* **Velocity**: Story points completed per sprint
* **Technical Debt**: Issues flagged for post-Vibathon cleanup

**Post-Vibathon Activities**

After the 48-hour development sprint, schedule:

1. **Retrospective Meeting**: Discuss what worked and what didn't
2. **Knowledge Transfer**: Document key decisions and implementation details
3. **Handover Documentation**: Create comprehensive documentation for maintenance
4. **Technical Debt Assessment**: Identify areas needing improvement
5. **Future Roadmap**: Plan next development phases
6. **Team Recognition**: Celebrate achievements and contributions

**Conclusion**

This team structure and role definition provides a solid foundation for a successful 48-hour Vibathon to implement the InterEd Admin Portal. By following this framework, the team can maximize productivity while maintaining code quality and meeting project objectives.

The key to success is clear communication, well-defined responsibilities, and a structured yet flexible approach that allows for rapid development without sacrificing quality. With proper preparation and organization, the team can achieve the ambitious goal of creating a comprehensive admin portal in just 48 hours.