**Tuition Central**

**System Overview**

The Tuition Central system follows a complete end-to-end workflow that connects requesters seeking tutoring services with qualified tutors. Here's how the epics work together to create a seamless experience:

**Main Workflow**

1. **User Registration and Authentication (Epic 1)**
   * Both requesters and tutors register and log in to the system
   * Authentication establishes user identity and role (requester or tutor)
2. **Requester Journey (Epic 2)**
   * Requester creates a tuition request specifying details like subject, education level, fee, location, and timeslots
   * Requester views their existing requests and selects an "open" request
   * Requester initiates a search for matching tutors
   * The Match-making System (Epic 5) finds and ranks suitable tutors
   * Requester reviews potential matches and sends offers to selected tutors
3. **Notification Process (Epic 4)**
   * System periodically checks for new offers (every 5 minutes)
   * Email notifications are sent to tutors about new offers
   * Offer status is updated to "notified" after sending
4. **Tutor Journey (Epic 3)**
   * Tutors create comprehensive profiles specifying subjects, education levels, experience, and fee expectations
   * Tutors receive and view offers in their dashboard
   * Tutors select and accept desired offers
   * System handles concurrency to ensure only one tutor can accept each request
5. **Request Completion**
   * When a tutor accepts an offer, the request status changes to "accepted"
   * Both requester and tutor are notified of the match
   * Contact details are exchanged to facilitate the tutoring arrangement

**System Integrations**

* The Match-making System (Epic 5) integrates with both Requester and Tutor functionalities to enable intelligent matching
* The Notification System (Epic 4) connects the offer creation process with tutor notifications
* All functionality is built on the foundation of User Authentication (Epic 1)

**Key Decision Points**

1. **Requester Decision Point**: After viewing matching tutors, the requester decides which tutors to send offers to
2. **Tutor Decision Point**: After receiving offers, the tutor decides which offers to accept
3. **System Decision Point**: The matching algorithm determines which tutors are suitable for each request

This workflow creates a balanced marketplace where requesters can find qualified tutors that match their specific needs, while tutors can find suitable teaching opportunities that align with their expertise and preferences.

**Epics and User Stories**

**Epic 1: User Authentication and Registration**

**User Story 1.1:** As a new user, I want to register with my email address so that I can create an account on Tuition Central.

* **Acceptance Criteria:**
  + Registration form includes fields for email address, full name, and password
  + User can select OIDC authentication with either Google or Facebook
  + System validates email format and checks for existing accounts
  + System generates a unique user ID
  + System captures full name and email address in Registration table
  + System sets user type field to "requester" or "tutor" based on registration path
  + User status is set to "active" upon successful registration
  + System sends a welcome email to verify the account
* **Technical Notes:**
  + Implement password hashing for security
  + Store OAuth tokens securely
  + Create database indexes on email and user ID fields
* **Dependencies:**
  + Integration with Google and Facebook OIDC services
  + Email service for verification

**User Story 1.2:** As a returning user, I want to log in with my credentials so that I can access my account.

* **Acceptance Criteria:**
  + Login form accepts email and password or OIDC authentication options
  + System validates credentials against stored information
  + System implements appropriate security measures (rate limiting, lockout after failed attempts)
  + System redirects user to the appropriate dashboard based on user type
  + System maintains session information securely
  + "Remember me" option available for convenience
  + Password reset functionality available
* **Technical Notes:**
  + Implement JWT or session-based authentication
  + Include CSRF protection
  + Log authentication attempts for security monitoring
* **Dependencies:**
  + User registration functionality
  + Integration with Google and Facebook OIDC services

**Epic 2: Requester Functionality**

**User Story 2.1:** As a requester, I want to create a tuition request so that I can find suitable tutors.

* **Acceptance Criteria:**
  + Form includes address field with optional geolocation assistance
  + Form includes calendar/time picker for selecting multiple preferred timeslots
  + Dropdown for education level selection (K1-2, P1-6, S1-5, J1-2)
  + Multi-select option for subjects (English, Chinese, Mathematics, Science, Chemistry, Physics, Biology)
  + Radio buttons or dropdown for tutor expertise level (Advanced, Good, Entry)
  + Numeric field with validation for fee willing to pay per hour
  + Date picker for selecting start date with appropriate constraints (no past dates)
  + Save button to store request in database with "open" status
  + System automatically assigns unique request ID
  + Confirmation message displayed upon successful submission
  + Draft saving functionality for incomplete requests
* **Technical Notes:**
  + Implement client-side validation for all fields
  + Add geolocation API integration for address input assistance
  + Create proper indexing in the database for search optimization
* **Dependencies:**
  + User authentication system
  + Request database schema

**User Story 2.2:** As a requester, I want to view all my tuition requests so that I can manage them.

* **Acceptance Criteria:**
  + Dashboard displays a paginated list of all requests created by the requester
  + Each request shows key information (subject, education level, status, date created)
  + Color-coding or icons indicate different statuses (open, accepted, completed)
  + Sorting options available (by date, status, etc.)
  + Filtering options available (by subject, status, etc.)
  + Detailed view available for each request
  + Edit functionality available for requests with "open" status
  + Cancel functionality available for requests with "open" status
  + Archive functionality for completed requests
* **Technical Notes:**
  + Implement optimized database queries for performance
  + Consider implementing lazy loading for better UI performance
* **Dependencies:**
  + Request creation functionality
  + Database schema for storing request status

**User Story 2.3:** As a requester, I want to search for matching tutors so that I can offer them tuition opportunities.

* **Acceptance Criteria:**
  + User can select an open request and click a "Look for Tutor" button
  + System performs matching based on specified criteria (education level, subject, fee, expertise level)
  + Results display tutor profiles with key information (name, expertise level, experience, fee range)
  + Results include profile completeness indicator
  + Pagination implemented for large result sets
  + Sorting options available (by expertise, fee, rating)
  + Additional filtering options available
  + Empty state messaging when no matches found
  + System provides suggestions to improve match results when few or no matches found
* **Technical Notes:**
  + Implement efficient search algorithm with appropriate indexes
  + Cache frequent search results for performance
  + Consider fuzzy matching for better results
* **Dependencies:**
  + Tutor profile creation functionality
  + Request creation functionality
  + Matching algorithm implementation

**User Story 2.4:** As a requester, I want to send offers to selected tutors so that I can engage their services.

* **Acceptance Criteria:**
  + Checkbox selection for multiple tutors from search results
  + "Send Offer" button becomes active only when at least one tutor is selected
  + Confirmation dialog before sending offers
  + Option to include additional message with the offer
  + System creates entries in Offer table with all required details
  + System sets offer status to "new"
  + Confirmation message displays number of offers sent
  + Offers appear in requester's "Sent Offers" section
  + Limit on maximum number of simultaneous offers per request
* **Technical Notes:**
  + Implement transaction handling to ensure data consistency
  + Create notification queue for processing
* **Dependencies:**
  + Tutor search functionality
  + Offer database schema

**Epic 3: Tutor Functionality**

**User Story 3.1:** As a tutor, I want to create my profile so that I can be matched with appropriate tuition requests.

* **Acceptance Criteria:**
  + Multi-select checkboxes for subject expertise (English, Chinese, Science, Mathematics, Chemistry, Physics, Biology)
  + Multi-select checkboxes for education levels taught (K1-2, P1-6, S1-4, J1-2)
  + Numeric field for years of experience with validation
  + Numeric field for number of students scoring A1 with validation
  + Range selector for expected fee per hour
  + Optional fields for qualifications, certifications, and teaching approach
  + Optional profile picture upload
  + Optional availability calendar
  + System automatically calculates expertise level based on A1 students (≤1: Entry, ≤3: Good, >3: Advanced)
  + Preview function before submission
  + Profile completion percentage indicator
  + Save and publish options
* **Technical Notes:**
  + Implement secure file upload for profile pictures
  + Create scheduled task to recalculate expertise levels
  + Consider versioning for profile updates
* **Dependencies:**
  + User registration system
  + File storage system for profile pictures

**User Story 3.2:** As a tutor, I want to view and manage offers so that I can accept tuition opportunities.

* **Acceptance Criteria:**
  + Dashboard displays all offers received with status filters (new, notified, accepted, declined)
  + Each offer shows complete request details (requester name, subject, education level, fee, location, timeslot, start date)
  + Distance/location information provided when location data is available
  + Map view option for visualizing offer locations
  + Sorting options (by date, fee, location)
  + Detailed view available for each offer
  + System distinguishes between new and previously viewed offers
  + Batch selection option for multiple offers
  + Option to request more information from requester
* **Technical Notes:**
  + Implement read receipts for offers
  + Consider geolocation services for distance calculation
  + Optimize queries for performance with large numbers of offers
* **Dependencies:**
  + Offer creation system
  + Requester offer sending functionality

**User Story 3.3:** As a tutor, I want to accept offers so that I can confirm tuition arrangements.

* **Acceptance Criteria:**
  + Checkbox selection for multiple offers
  + "Accept" button activates only for selected offers
  + Confirmation dialog before acceptance
  + System attempts to update request status to "accepted" if still "open"
  + System updates offer status to "accepted" when successful
  + System provides clear feedback for each offer (accepted, already taken)
  + Option to include message with acceptance
  + System records acceptance timestamp
  + Notification sent to requester upon acceptance
  + Contact information exchange facilitated after acceptance
* **Technical Notes:**
  + Implement database transactions to ensure data consistency
  + Handle race conditions for simultaneous acceptances
  + Include audit logging for acceptance actions
* **Dependencies:**
  + Offer viewing functionality
  + Notification system

**Epic 4: Notification System**

**User Story 4.1:** As a system administrator, I want the system to send notifications to tutors about new offers so that they can respond promptly.

* **Acceptance Criteria:**
  + Background service runs every 5 minutes to check for new offers
  + Service identifies offers with status "new"
  + System generates appropriate email content with offer details
  + System sends personalized email notifications to tutors
  + Email includes direct link to the specific offer
  + System updates offer status to "notified" after sending
  + System logs notification attempts and successes/failures
  + Retry mechanism for failed notifications
  + Rate limiting to prevent email flooding
  + Configurable notification schedule and frequency
* **Technical Notes:**
  + Implement message queue for notification processing
  + Create email templates for consistent formatting
  + Configure monitoring for the notification service
* **Dependencies:**
  + Email service integration
  + Offer database schema

**User Story 4.2:** As a tutor, I want to receive notifications about new offers so that I can respond to them.

* **Acceptance Criteria:**
  + Tutor receives email notification containing:
    - Requester information (name only, no personal contact details)
    - Subject and education level required
    - Location area (general, not specific address)
    - Fee offered
    - Preferred timeslots
    - Starting date
  + Email contains a prominent call-to-action button
  + Direct link to the offer in the portal
  + Option to accept offer directly from email (with security token)
  + Option to manage notification preferences (frequency, channel)
  + In-app notifications in addition to email
  + Notification center in the application for message history
  + Option to enable/disable different notification types
* **Technical Notes:**
  + Implement secure tokens for direct actions from email
  + Consider push notifications for mobile users
  + Track email open and click rates
* **Dependencies:**
  + Notification sending system
  + User preference storage

**Epic 5: Match-making System**

**User Story 5.1:** As a system administrator, I want the system to match requesters with tutors based on specified criteria so that appropriate connections are made.

* **Acceptance Criteria:**
  + System implements matching algorithm considering:
    - Exact education level match
    - Subject match (single or multiple)
    - Fee compatibility (within tutor's expected range)
    - Expertise level compatibility
    - Optional proximity/location matching when data available
    - Optional availability matching when data available
  + Algorithm weights factors appropriately
  + Matches sorted by relevance score
  + System returns paginated, sorted matching tutor profiles
  + Performance metrics tracked (response time, match quality)
  + System handles edge cases (no matches, too many matches)
  + Algorithm can be tuned and configured without code changes
  + Matching statistics available for system administrators
* **Technical Notes:**
  + Implement efficient indexing for search performance
  + Create caching mechanism for frequent searches
  + Design for horizontal scaling as user base grows
* **Dependencies:**
  + Tutor profile data
  + Request data
  + Performance monitoring system

**User Story 5.2:** As a requester or tutor, I want to see match quality indicators so that I can make informed decisions.

* **Acceptance Criteria:**
  + System calculates and displays match percentage or score
  + Visual indicators show strength of match for different criteria
  + Explanation provided for why a match was made
  + Mismatches or compromises clearly highlighted
  + Suggestions provided for improving match quality
  + Historical match success rates displayed when available
  + Rating system for matches after completion
* **Technical Notes:**
  + Create algorithm for calculating match quality
  + Design intuitive visual indicators
  + Implement feedback loop for improving match algorithm
* **Dependencies:**
  + Matching algorithm
  + User interface components for displaying match quality

This expanded set of user stories provides much more detail to guide development and ensure all requirements are properly addressed. Each story now includes comprehensive acceptance criteria that specify exactly what functionality is needed, along with technical notes and dependencies to assist in implementation planning.