Citizen Developer

Posting Board

Technical Documentation

System Launch Edition

Version 1.0 - Launch Release
Generated: July 26, 2025

Table of Contents

1. Executive Summary

1.1 System Overview

The Citizen Developer Posting Board represents a strategic initiative to democratize software development within our organization. This platform creates a marketplace where business teams can post their automation and development needs, while technically-skilled employees from any department can discover and implement these solutions. By leveraging the growing citizen developer movement, this system addresses the critical gap between technical capacity and business demand for digital solutions. The platform facilitates collaboration, skill development, and rapid solution delivery while maintaining appropriate governance and oversight.

1.2 Launch Objectives

The system launch aims to achieve the following key objectives:

- Establish a centralized platform for capturing and prioritizing business automation needs across all departments
- Enable skilled employees to contribute to digital transformation efforts regardless of their formal role
- Reduce the backlog of small to medium-sized development requests through distributed implementation
- Create visibility into the organization's collective technical capabilities and development capacity
- Foster a culture of innovation and continuous improvement through democratized problem-solving
- Implement appropriate governance and approval workflows for citizen development initiatives

1.3 Expected Benefits

Based on industry benchmarks and pilot program results, we anticipate the following benefits within the first year of operation:

- 30-40% reduction in development backlog for small automation and enhancement requests
- Identification and enablement of 50+ citizen developers across the organization
- Average 2-3 week reduction in time-to-solution for qualifying projects

- Improved employee engagement through skill development and contribution opportunities
- Enhanced visibility into departmental technology needs and priorities
- Creation of a knowledge base of reusable solutions and best practices

1.4 Success Metrics

The platform will track the following key performance indicators to measure success and guide continuous improvement:

Metric	Target (Year 1)	Measurement Method
Active Citizen Developers	50+ users	Unique users claiming ideas
Ideas Submitted	200+ ideas	Total submissions in system
Ideas Completed	100+ solutions	Ideas marked as complete
Average Time to Claim	< 5 days	Time from submission to claim
Average Completion Time	< 30 days	Time from claim to completion
User Satisfaction	> 80%	Quarterly survey results

The Citizen Developer Posting Board positions our organization at the forefront of the digital transformation movement. By empowering our workforce to participate directly in solution development, we create a sustainable competitive advantage through increased agility, innovation, and employee engagement. This platform serves as the foundation for a broader citizen development program that will evolve based on user feedback and organizational needs.

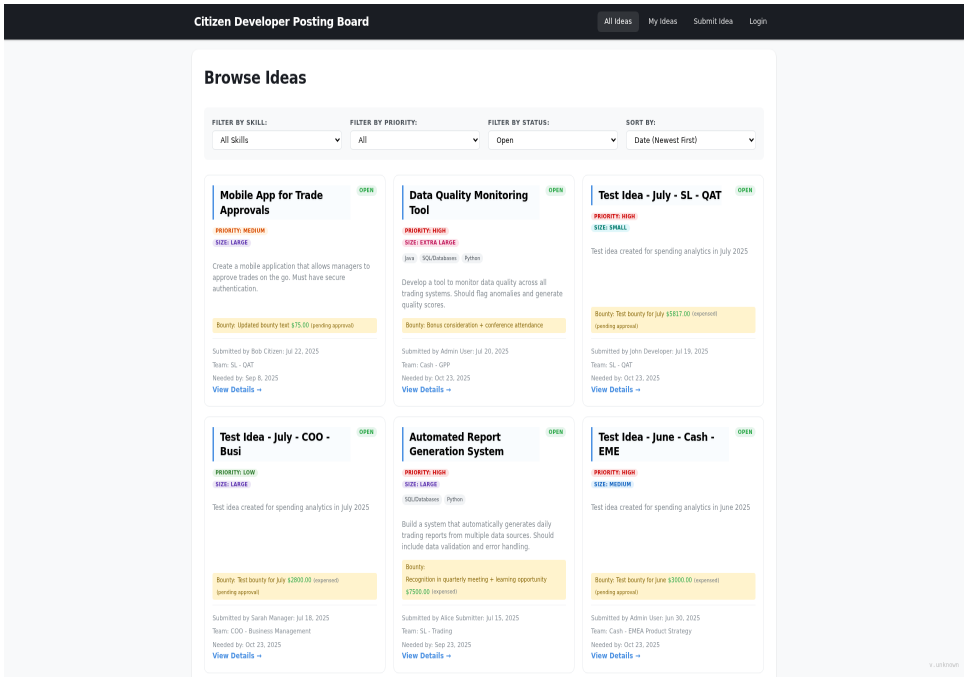
2. Technical Architecture

2.1 Technology Stack

The platform is built on a modern, maintainable technology stack that prioritizes developer productivity, system reliability, and operational simplicity:

Layer	Technology	Purpose
Backend Framework	Flask 2.3+	Lightweight Python web framework for rapid development
Database	SQLite/PostgreSQL	Embedded database for development, PostgreSQL for production
ORM	SQLAlchemy	Database abstraction layer with migration support
Frontend	Jinja2 + JavaScript	Server-side templating with dynamic client interactions
Session Management	Flask-Session	Secure server-side session storage with Redis support
Authentication	Custom Email-based	Passwordless authentication for enhanced security
Deployment	Docker + Gunicorn	Containerized deployment with production WSGI server
Version Control	Git	Source code management and collaboration

System Homepage



2.4 Session Management

The platform implements a robust session management system that maintains user state across requests while ensuring security and performance. Flask-Session provides server-side session storage, preventing client-side tampering and enabling complex session data structures.

Session Configuration

```
app.config['SESSION_TYPE'] = 'filesystem' app.config['SESSION_PERMANENT'] = False
app.config['PERMANENT_SESSION_LIFETIME'] = timedelta(days=7)
app.config['SESSION_FILE_DIR'] = './flask_session/' app.config['SESSION_COOKIE_SECURE'] =
True # HTTPS only in production app.config['SESSION_COOKIE_HTTPONLY'] = True # Prevent
XSS attacks app.config['SESSION_COOKIE_SAMESITE'] = 'Lax' # CSRF protection
```

Session Variables Reference

The following session variables are used throughout the application to maintain user state and provide personalized experiences:

user_email (String (required))

The authenticated user's email address, serving as the primary identifier throughout the system. This value is set during the email verification process and persists for the duration of the session. Used for database lookups, audit trails, and access control. Format: Standard email format (e.g., user@company.com). Maximum length: 120 characters.

```
Example: john.doe@company.com
```

user_name (String (optional))

The user's display name as entered in their profile. This human-readable name appears throughout the UI instead of the email address for better user experience. If not set, the system falls back to displaying the email address. Updated when users complete their profile or modify their name. Maximum length: 100 characters.

```
Example: John Doe
```

user_verified (Boolean (required))

Indicates whether the user has successfully completed email verification. Set to True after entering a valid verification code. This flag gates access to protected features like

submitting ideas, claiming ideas, and accessing personal dashboards. Unverified users can only browse public content. Reset to False if verification expires.

```
Example: True
```

user_role (String Enum (required))

The user's assigned role within the system, determining their permissions and available features. Valid values: 'manager' (can approve claims and view team data), 'developer' (can claim and implement ideas), 'citizen_developer' (same as developer), 'idea_submitter' (can only submit ideas). Set during profile creation and modifiable by administrators only.

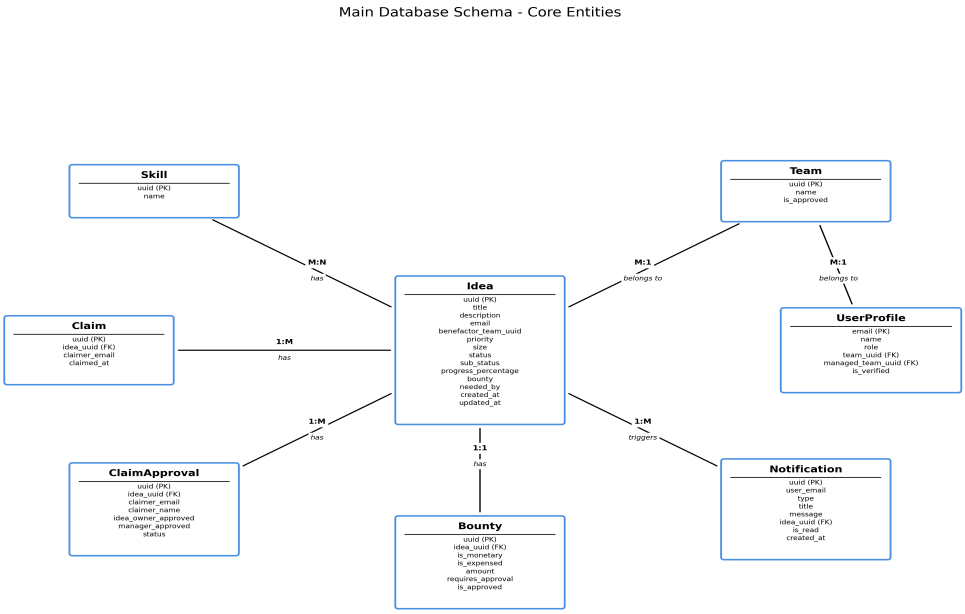
```
Example: developer
```

3. Database Design

3.1 Core Schema

The database follows a normalized design with UUID primary keys for enhanced security and scalability. All tables use 36-character UUID strings as primary keys, preventing enumeration attacks and supporting distributed systems. The schema is designed for flexibility and performance with appropriate indexes on foreign keys and commonly queried fields.

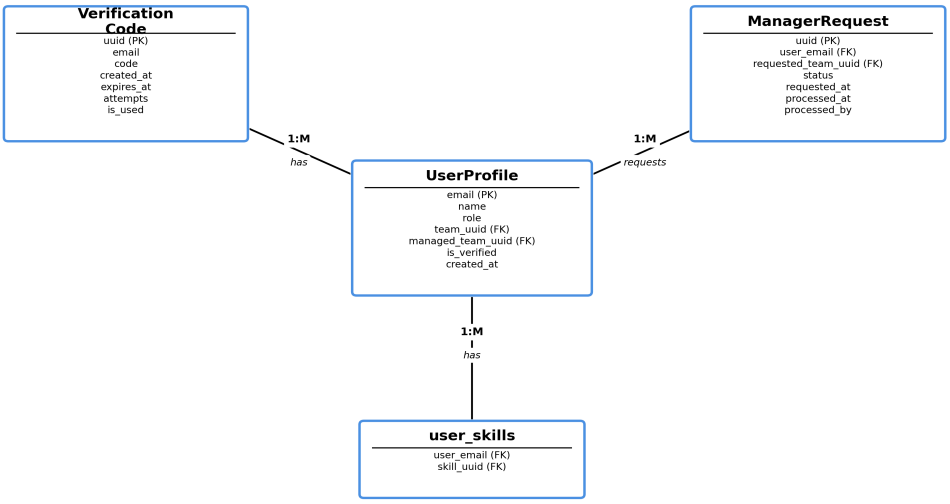
Entity Relationship Diagram - Core Tables



3.2 Entity Relationships

Authentication and User Management Tables

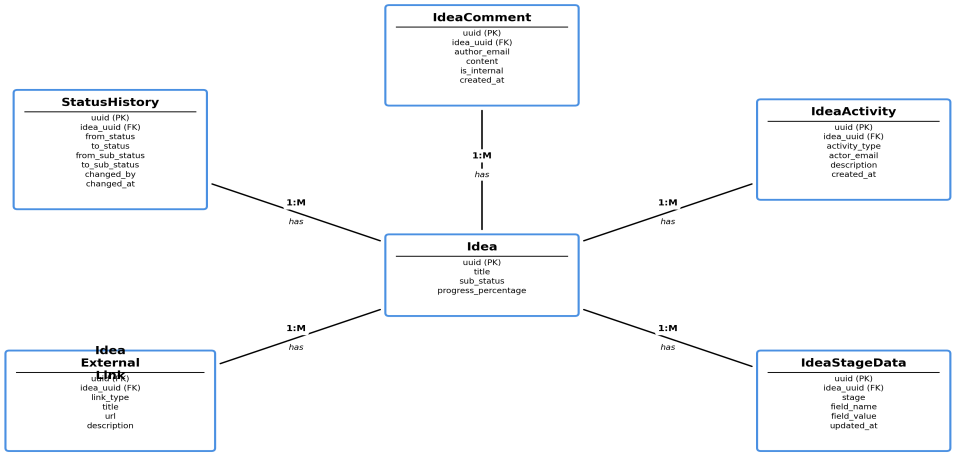
Authentication & User Management Schema



3.4 SDLC Extensions

Software Development Lifecycle Tables

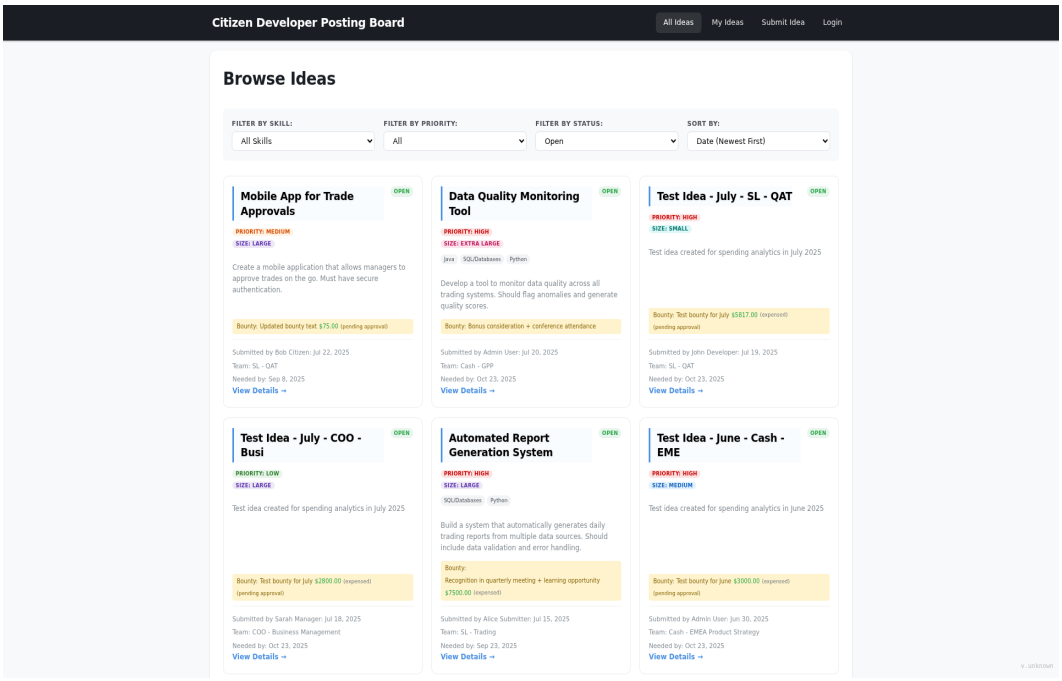
SDLC Tracking Schema



4. User Interface

4.1 Browse Ideas

The home page provides a comprehensive view of all available ideas with advanced filtering and sorting capabilities. Users can quickly identify opportunities that match their skills and interests.

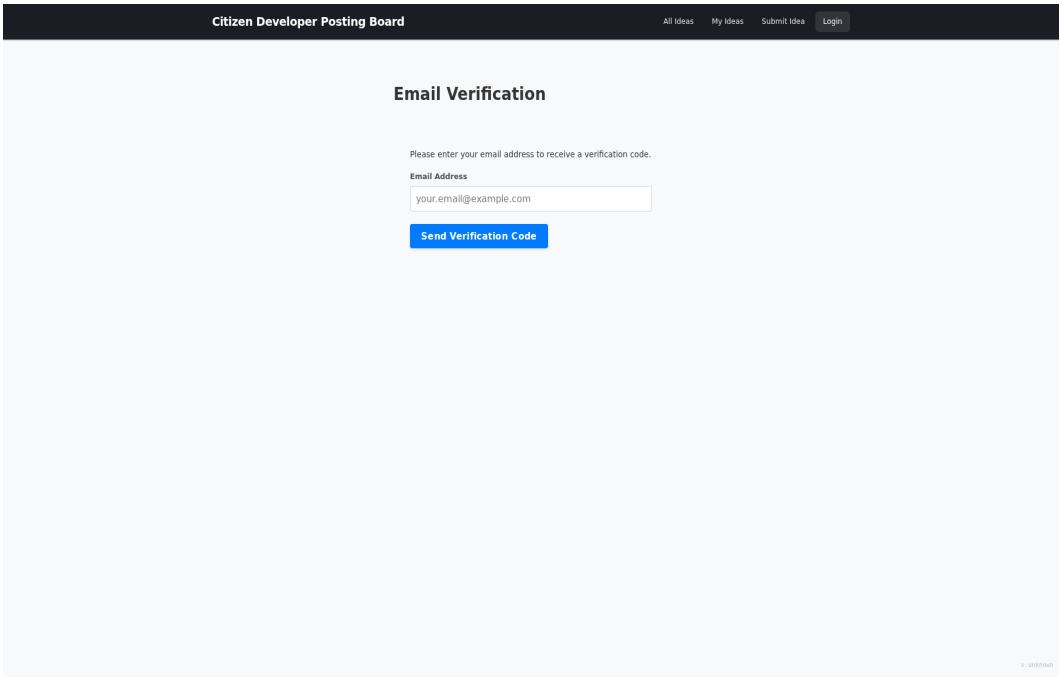


Key Features:

- Real-time filtering by skill, priority, status, and team
- Visual indicators for idea priority and size
- Bounty information prominently displayed
- One-click access to detailed idea information
- Responsive grid layout adapting to screen size
- Auto-refresh to show latest submissions

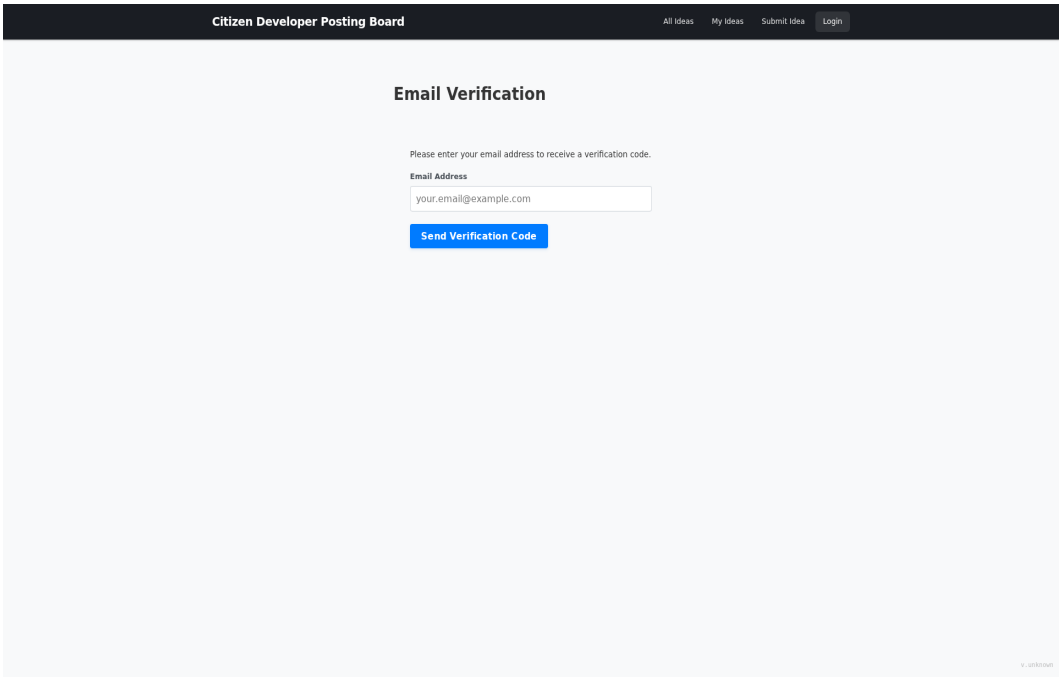
4.2 Submit Ideas

The idea submission interface provides a streamlined process for users to capture their automation and development needs with all necessary context.


Citizen Developer Posting Board
All Ideas
My Ideas
Submit Idea
Login
Email Verification
Please enter your email address to receive a verification code.
Email Address
your.email@example.com
Send Verification Code
v.-unknown

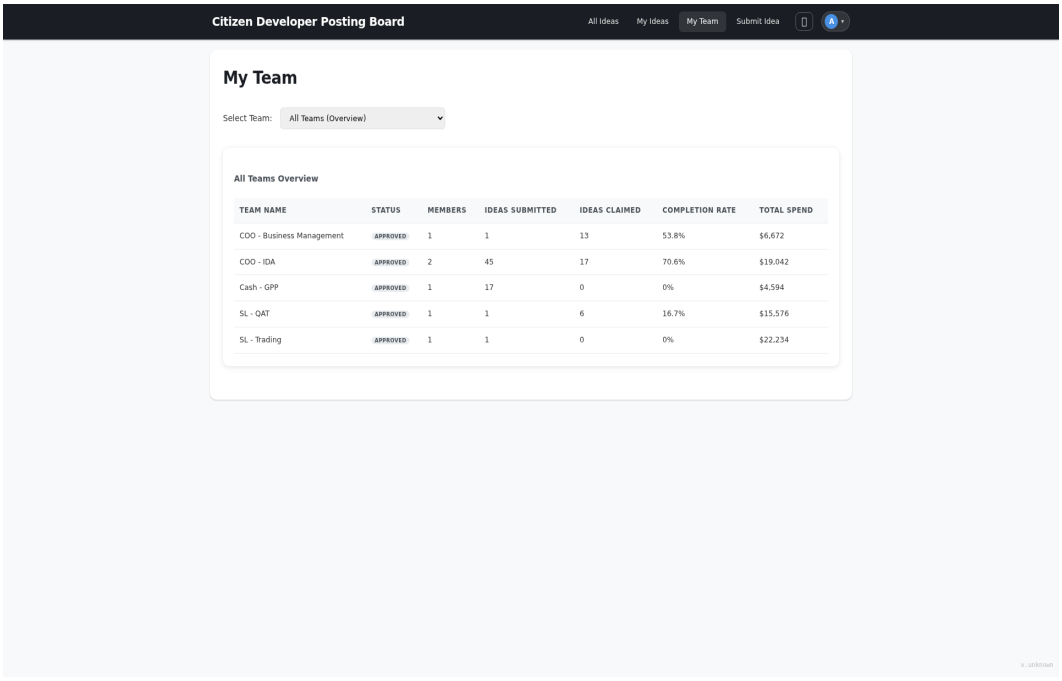
4.3 My Ideas Dashboard

The My Ideas page provides a personalized view of all ideas submitted by or claimed by the authenticated user, including pending approvals and activity tracking.



4.4 Team Analytics

The My Team page provides comprehensive analytics and management capabilities for team managers, including performance metrics, skill gap analysis, and spending tracking.

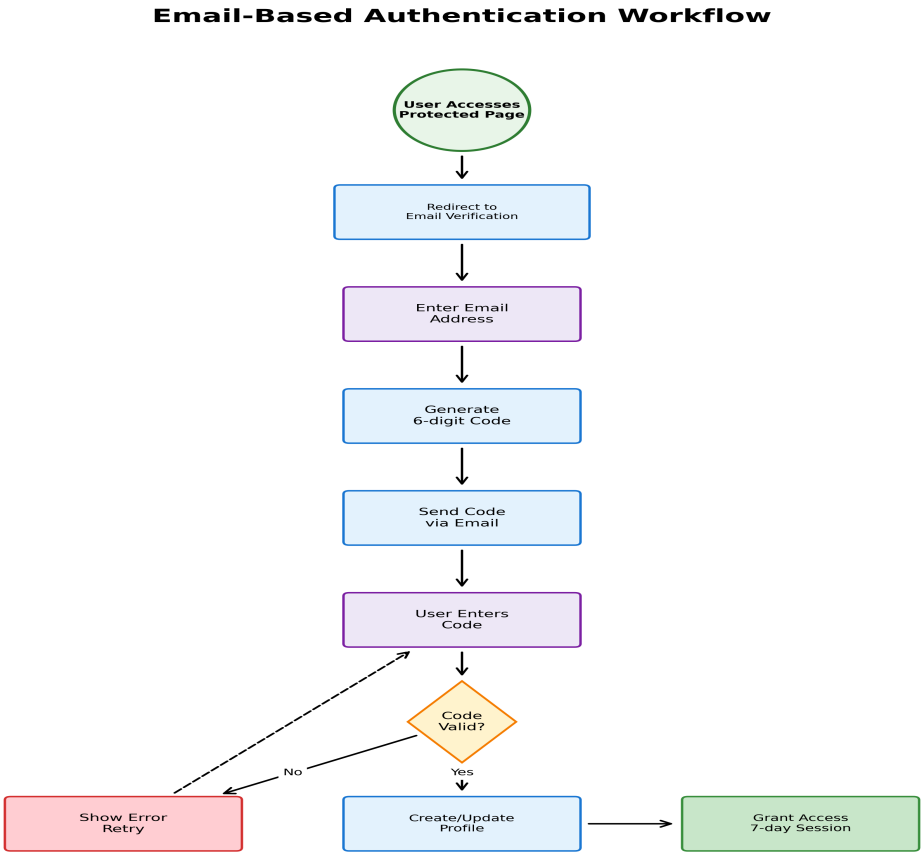


5. Core Workflows

5.1 Authentication Flow

The platform implements a passwordless authentication system that balances security with user convenience. This approach eliminates password-related vulnerabilities while providing a smooth user experience.

Authentication Workflow Diagram



Authentication Steps:

1. User enters email address on verification page
2. System generates 6-digit verification code
3. Code sent via email (or displayed in console for development)
4. User enters code within 3-minute expiration window
5. System validates code and creates authenticated session

6. User redirected to complete profile if first-time login
7. Session persists for 7 days with sliding expiration

5.3 Claim Process

The claim process implements a dual-approval workflow ensuring both the idea owner and the claimer's manager approve before work begins. This provides appropriate oversight while maintaining agility.

Claim Approval Workflow

6. API Documentation

6.1 RESTful Design

The platform provides a comprehensive RESTful API following industry best practices for naming conventions, HTTP methods, status codes, and response formats. All endpoints return JSON responses with consistent structure and error handling.

API Design Principles:

- Resource-based URLs using nouns, not verbs
- HTTP methods indicate actions (GET, POST, PUT, DELETE)
- Consistent JSON response format with success indicators
- Detailed error messages with actionable information
- UUID-based resource identification for security
- Pagination support for list endpoints
- Filtering and sorting via query parameters

6.2 Public Endpoints

These endpoints are accessible without authentication:

GET /api/ideas

Retrieves a filtered list of ideas based on query parameters. Returns all ideas by default, or filtered by status, priority, skill, or team. Supports sorting by date, priority, or alphabetical order.

Parameters:

- skill (optional): Filter by required skill name
- priority (optional): Filter by priority level (low, medium, high)
- status (optional): Filter by status (open, claimed, complete)
- benefactor_team (optional): Filter by team UUID
- sort (optional): Sort order (date_desc, date_asc, priority, alphabetical)

Example Response:


```
{
  "success": true,
  "ideas": [
    {
```

```
      "id": "uuid-string",
      "title": "Idea title",
      "description": "Detailed description",
      "priority": "medium",
      "size": "large",
      "status": "open",
      "skills": ["Python", "SQL"],
      "bounty": "Recognition in team meeting",
      "bounty_details": {
        "is_monetary": true,
        "amount": 500.00,
        "is_expensed": true,
        "is_approved": false
      },
      "team_name": "SL - Tech",
      "submitter_name": "John Doe",
      "created_at": "2025-01-26",
      "needed_by": "2025-02-15"
    }
  ]
}
```

GET /api/skills

Returns a list of all available skills in the system. Used for populating filter dropdowns and skill selection interfaces.

Example Response:

```
[
  {
    "id": "uuid-string",
    "name": "Python",
    "category": "Programming"
  },
  {
    "id": "uuid-string",
    "name": "SQL",
    "category": "Database"
  }
]
```

7. Security and Authentication

7.1 Passwordless Authentication

The platform implements a passwordless authentication system that eliminates common password-related vulnerabilities while maintaining strong security. This approach uses time-limited verification codes sent via email.

Security Features:

- No password storage eliminates breach risks
- 6-digit verification codes with 3-minute expiration
- Rate limiting prevents brute force attempts (3 attempts per hour)
- Email validation ensures valid corporate addresses
- Session tokens use cryptographically secure generation
- HTTPOnly cookies prevent XSS attacks
- CSRF protection via SameSite cookie attribute

8. SDLC Features

8.1 Sub-Status Tracking

The platform includes comprehensive Software Development Life Cycle (SDLC) tracking features that enable detailed monitoring of idea progress through development stages. These features provide transparency and accountability throughout the implementation process.

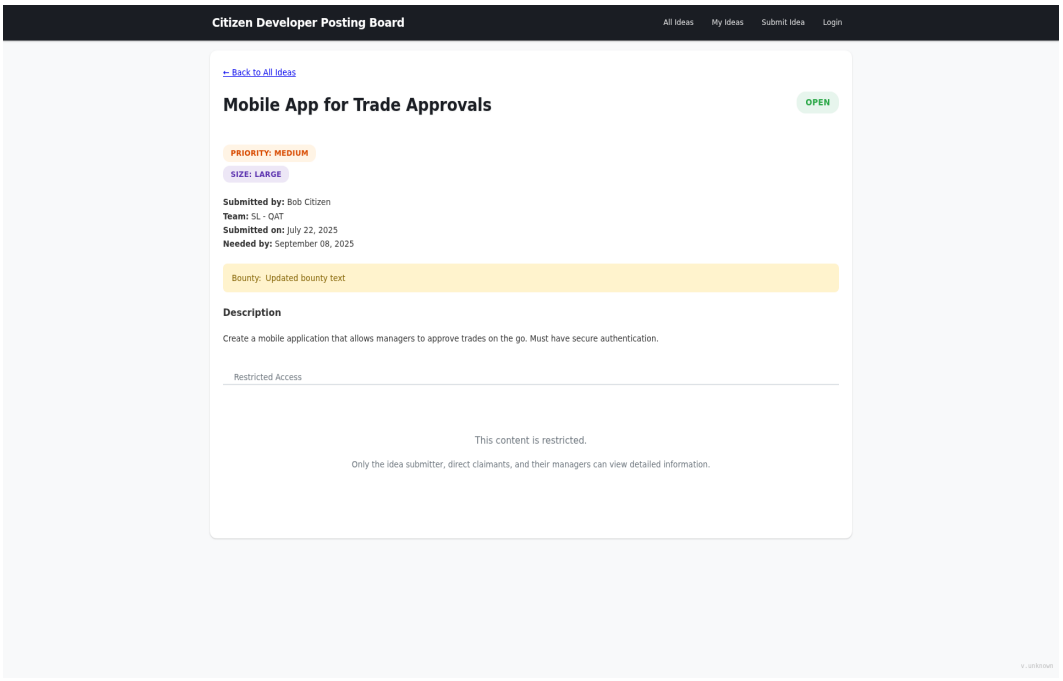
Development Sub-Statuses:

- planning: Initial requirements gathering and design (10% progress)
- in_development: Active development work (30% progress)
- testing: Quality assurance and testing (60% progress)
- awaiting_deployment: Ready for production deployment (80% progress)
- deployed: Deployed to production environment (90% progress)
- verified: Fully tested and verified in production (100% progress)
- on_hold: Temporarily paused (maintains current progress)
- blocked: Blocked by dependencies or issues (maintains progress)
- cancelled: Development cancelled (0% progress)
- rolled_back: Deployment rolled back (0% progress)

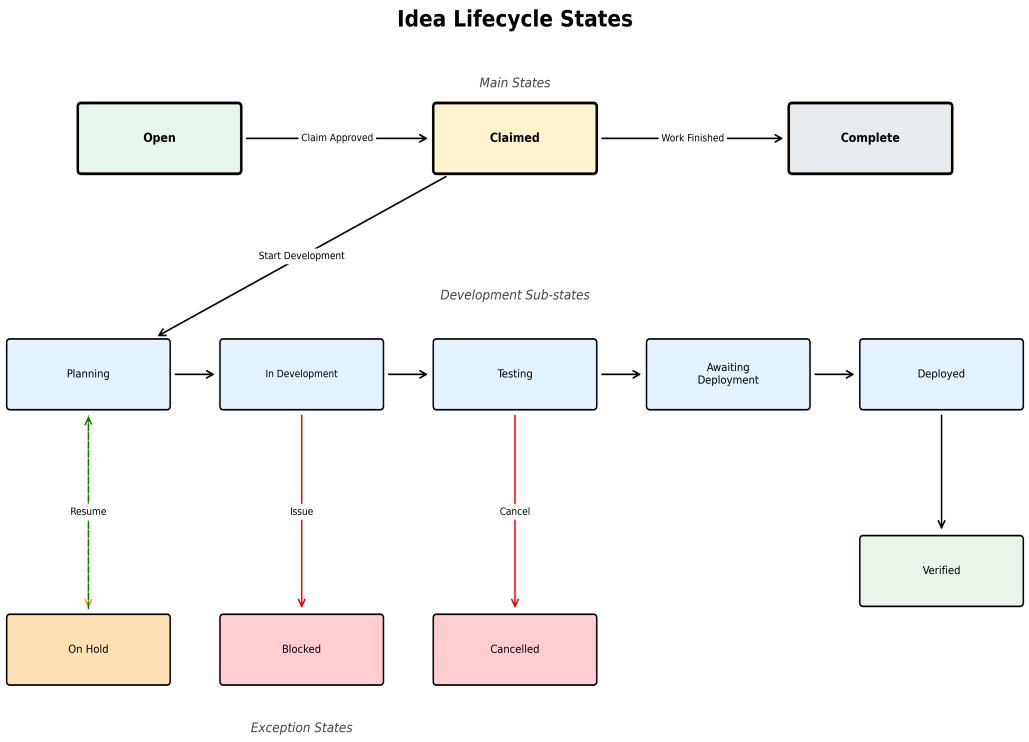
8.2 Development Progress

The platform provides visual progress tracking through interactive GANTT charts and progress indicators, giving stakeholders real-time visibility into development status.

Idea Detail with SDLC Tracking



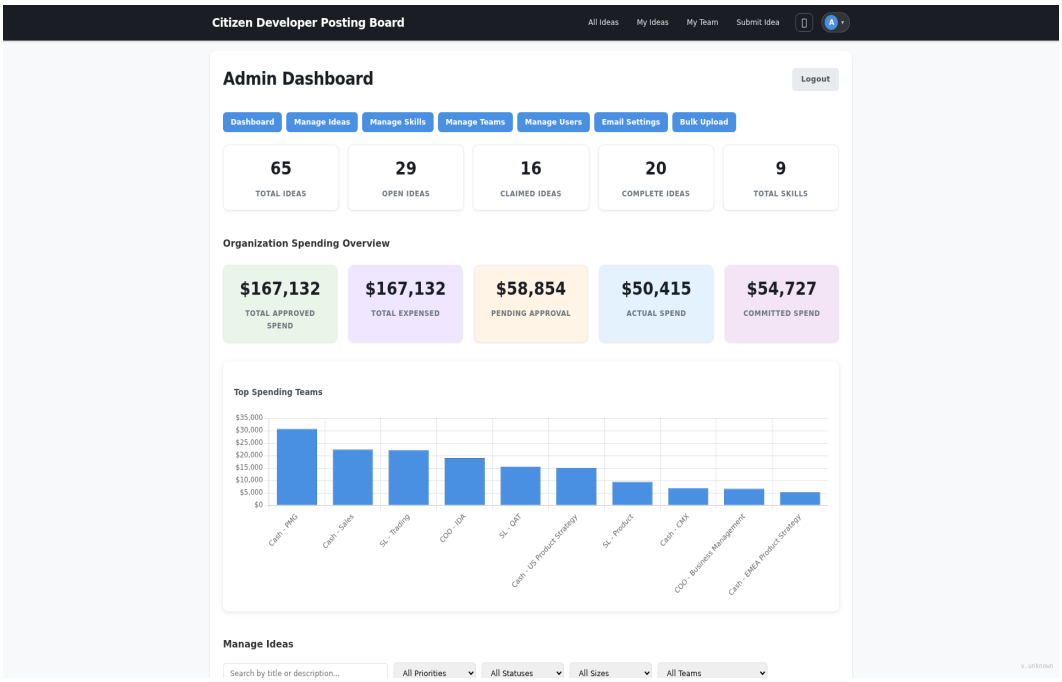
Idea Lifecycle Flow



9. Admin Portal

9.1 Dashboard Overview

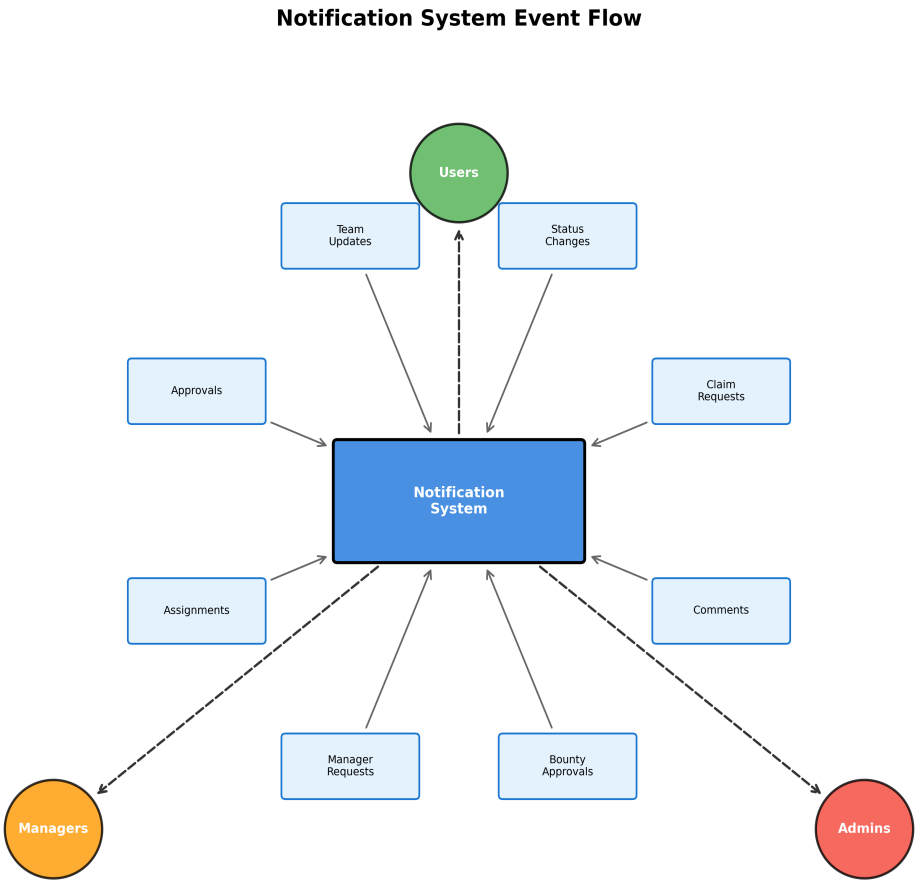
The administrative portal provides comprehensive system management capabilities with real-time analytics, user management, and configuration options. Access is restricted to authorized administrators through additional authentication.



Key Metrics Displayed:

- Total ideas submitted across all teams
- Open ideas awaiting claims
- Active claims in progress
- Completed ideas with success metrics
- Total registered users and skill distribution
- Spending analytics for monetary bounties
- Team performance comparisons

Notification System Workflow



10. Deployment Guide

10.1 Requirements

The platform is designed for easy deployment in various environments with minimal dependencies. Both development and production configurations are supported.

System Requirements:

- Python 3.8 or higher (3.12 recommended)
- 2GB RAM minimum (4GB recommended)
- 10GB disk space for application and data
- Linux or macOS for production (Windows supported for development)
- PostgreSQL 12+ for production database (SQLite for development)
- SMTP server access for email notifications
- HTTPS certificate for production deployment

11. Performance Optimization

11.1 Database Optimization

The platform implements several optimization strategies to ensure responsive performance even with large datasets and concurrent users.

Optimization Techniques:

- UUID indexes on all foreign key columns
- Composite indexes for common query patterns
- Query result caching for frequently accessed data
- Lazy loading of related entities
- Connection pooling for database connections
- Prepared statements to prevent SQL injection
- Pagination for large result sets

12. Troubleshooting

12.1 Common Issues

This section addresses frequently encountered issues and their solutions to help administrators and developers quickly resolve problems.

Issue	Cause	Solution
Email verification codes not received	SMTP configuration incorrect or email server blocking	Check SMTP settings in config, verify firewall rules, check spam folder
Session expires unexpectedly	Session timeout too short or cookie settings incorrect	Adjust SESSION_LIFETIME in config, verify cookie domain settings
Cannot claim ideas	User profile incomplete or wrong role assigned	Ensure user has developer/citizen_developer role and skills selected
Admin portal access denied	Admin session expired or incorrect password	Re-authenticate with admin password, check session configuration

13. Future Roadmap

13.1 Planned Features

The following features are planned for future releases based on user feedback and organizational needs:

- Microsoft Teams integration for notifications
- AI-powered idea matching and recommendations
- Mobile application for iOS and Android
- Advanced analytics with PowerBI integration
- Automated testing framework for citizen developers
- Template library for common solutions
- Gamification elements to encourage participation
- Integration with corporate SSO systems
- Multi-language support for global teams
- API webhooks for external integrations

14. Appendices

14.1 Configuration Reference

The following configuration options are available for customizing the platform:

DATABASE_URL:
PostgreSQL connection string for production

SECRET_KEY:
Secret key for session encryption (generate unique value)

SMTP_SERVER:
Email server hostname for notifications

SMTP_PORT:
Email server port (typically 587 for TLS)

SMTP_USERNAME:
Email account username

SMTP_PASSWORD:
Email account password

SESSION_LIFETIME:
Session duration in seconds (default: 604800 = 7 days)

VERIFICATION_CODE_EXPIRY:
Code expiration in seconds (default: 180 = 3 minutes)

BOUNTY_APPROVAL_THRESHOLD:
Amount requiring approval (default: 50)

DEBUG:
Enable debug mode (never true in production)