📋 ASCEND - Product Requirements Document (PRD)

📌 Document Information

Field	Details
Product Name	ASCEND (Alumni–Student Career ENgagement Dashboard)
Version	1.0
Date	January 25, 2026
Author	Product Team
Status	Draft for Development
Confidentiality	Internal Use Only

📑 Table of Contents

---

1. Executive Summary

1.1 Product Vision

ASCEND is a college-specific alumni-student mentorship platform that revolutionizes career guidance by connecting junior students with seniors and alumni working at specific companies. Unlike generic mentorship platforms, ASCEND enables students to discover mentors based on their target companies (e.g., Odoo, Google, TCS) and receive hyper-targeted, company-specific career guidance, interview preparation, roadmaps, and referral opportunities through a structured, asynchronous mentorship model.

1.2 Problem Statement

Current Challenges:

Junior students in higher education institutions face significant barriers when seeking career guidance from experienced professionals:

- **Lack of Structured Access:** No centralized platform to discover which seniors/alumni work at specific target companies
- **Low Response Rates:** Cold messaging on LinkedIn yields <10% response rate; WhatsApp groups are chaotic and unstructured
- **Generic Advice:** Career counseling is too broad and not tailored to specific companies or roles
- **No Accountability:** No way to verify quality of advice or track outcomes
- **Mentor Burnout:** Alumni get overwhelmed with repetitive questions and random messages
- **Missing Company Context:** Students can't find insider information about company culture, interview processes, or preparation strategies
- **No Referral Pipeline:** Informal networks make it difficult to request and track referrals

Impact: These issues result in inefficient knowledge transfer, poor career decision-making, missed opportunities, and underutilization of valuable alumni networks.

1.3 Proposed Solution

ASCEND addresses these challenges through:

1. **Company-Based Discovery:** Students search mentors by target company, not just skills
2. **Asynchronous Structured Mentorship:** Question-answer model (not real-time chat) with mandatory context fields
3. **Fair Queue Management:** FIFO-based distribution prevents mentor overload using Data Structures
4. **Outcome-Based Trust System:** Quality control through delayed feedback on advice effectiveness
5. **Intelligent Matching:** Algorithm matches students with most relevant mentors based on company, skills, and availability
6. **Referral Workflow:** Formal pipeline from guidance → portfolio building → referral request
7. **Knowledge Base:** Searchable archive of answered questions reduces duplicate queries

1.4 Key Differentiators

Aspect	**Traditional Platforms**	**ASCEND**
Discovery	By skills/industry	By specific company
Communication	Real-time chat/calls	Async structured Q&A
Network	Global/open	College-specific (trusted)
Quality Control	Ratings only	Outcome-based trust scores
Mentor Load	Uncontrolled	Queue-managed (DSA-based)
Company Intel	Generic	Aggregated insider insights
Referrals	Informal	Structured workflow

1.5 Success Criteria

6-Month Post-Launch Targets:

- **User Adoption:** 500+ active students, 100+ verified alumni
- **Engagement:** 1,000+ questions answered, 70%+ mentor response rate
- **Quality:** Average response time <3 days, student satisfaction >4.2/5
- **Outcomes:** 20+ successful referrals, 10+ job placements attributed to platform

- **Knowledge Base:** 500+ searchable Q&A entries reducing duplicate questions by 40%

---

2. Product Overview

2.1 Product Description

ASCEND is a web-based platform that creates a structured bridge between junior students and college alumni/seniors working at specific companies. The platform enables students to:

- Discover which seniors/alumni work at their target companies
- Submit structured questions about company-specific career guidance
- Receive expert responses from verified professionals
- Access aggregated company intelligence (hiring process, culture, tech stack)
- Build portfolios and request referrals through formal workflows
- Learn from a searchable archive of previously answered questions

2.2 Target Market

Primary Market: Engineering colleges and universities in India with active alumni networks

Initial Launch: DDU (Dharmsinh Desai University) or similar tier-2/tier-3 engineering colleges

Expansion Potential:

- Other departments (MBA, Science, Commerce)
- Other colleges in the region
- Pan-India engineering college network

2.3 Product Type

Platform: Web Application (Responsive)

Access: Browser-based (Desktop & Mobile)

Deployment: Cloud-hosted (scalable architecture)

Business Model: Free for students and alumni (college-sponsored or freemium model)

2.4 Core Value Propositions

For Students:

- Find mentors at target companies (Odoo, Google, TCS, etc.)
- Get company-specific roadmaps and interview tips

- Access insider knowledge (culture, hiring, salary ranges)
- Build credibility for referral requests
- Learn from archived Q&A (knowledge base)

For Alumni:

- Give back to college community efficiently
- Build personal brand as mentor/expert
- Stay connected with college network
- Control mentorship workload (set limits)
- Potential recruitment pipeline for their companies

For College Administration:

- Improve student placement rates
- Engage alumni network productively
- Track mentorship outcomes and success stories
- Showcase placement success for accreditation
- Build institutional knowledge repository

---

3. Goals and Objectives

3.1 Primary Goals

Goal 1: Create Structured Alumni-Student Connections

- Enable discovery of alumni by target company
- Facilitate meaningful, context-rich conversations
- Build trust through verified profiles

Goal 2: Deliver High-Quality Career Guidance

- Ensure company-specific, actionable advice
- Maintain quality through outcome-based feedback
- Reduce generic/low-value responses

Goal 3: Prevent Mentor Burnout

- Implement fair question distribution (queue system)

- Allow mentors to set availability limits
- Filter out low-quality/duplicate questions

Goal 4: Build Institutional Knowledge

- Create searchable archive of Q&A
- Aggregate company intelligence from alumni
- Document success stories and placement journeys

Goal 5: Facilitate Career Outcomes

- Enable structured referral requests
- Track student progress toward target companies
- Measure placement success attributed to platform

3.2 Measurable Objectives

User Acquisition (6 months):

- Register 500+ students (50% of eligible junior students)
- Onboard 100+ verified alumni (targeting 20% of recent graduates)
- Achieve 70% monthly active user rate among registered students

Engagement Metrics:

- 1,000+ questions submitted
- 70%+ question response rate from mentors
- Average response time <3 days
- 40% reduction in duplicate questions (via knowledge base)

Quality Metrics:

- Student satisfaction rating >4.2/5
- Mentor trust score average >75/100
- 60%+ of students report "advice was actionable"

Outcome Metrics:

- 20+ successful referral requests processed
- 10+ students placed at target companies via platform
- 5+ success stories documented and approved

Academic Integration:

- Demonstrate practical application of 5+ Data Structures and Algorithms
- Complete project within single semester timeline
- Achieve modular, scalable codebase for future extensions

---

4. Target Users

4.1 User Segments

Primary Users

1. Junior Students (Year 1-3)

- **Demographics:** 18-21 years old, undergraduate engineering students
- **Technical Proficiency:** Basic to intermediate (comfortable with web applications)
- **Needs:** Career direction, interview preparation, company insights, skill development roadmaps, referrals
- **Pain Points:**
    - Don't know which seniors work at target companies
    - Fear of bothering alumni with cold messages
    - Receive generic advice that doesn't apply to specific companies
    - No structured way to prepare for target company interviews
    - Lack credibility when requesting referrals
- **Goals:**
    - Secure internships/jobs at preferred companies
    - Build relevant skills efficiently
    - Get insider interview tips
    - Obtain referrals from trusted sources

2. Senior Students & Recent Alumni (0-5 years experience)

- **Demographics:** 21-26 years old, final year students or recent graduates
- **Technical Proficiency:** Intermediate to advanced
- **Needs:** Give back to college, build personal brand, stay connected

- **Pain Points:**
    - Get spammed with messages on LinkedIn/WhatsApp
    - No time for lengthy phone calls or meetings
    - Repetitive questions drain energy
    - Can't track impact of their mentorship
- **Goals:**
    - Help juniors efficiently (limited time commitment)
    - Maintain college network
    - Build reputation as helpful mentor
    - Potentially identify recruitment candidates

3. Experienced Alumni (5+ years experience)

- **Demographics:** 26-35 years old, mid-level to senior professionals
- **Technical Proficiency:** Advanced
- **Needs:** Mentorship as professional development, legacy building
- **Pain Points:**
    - Very limited time availability
    - Want to help but need structured approach
    - Difficult to assess which students are serious
- **Goals:**
    - Mentor select serious students
    - Share career wisdom efficiently
    - Identify high-potential candidates for their companies
    - Build leadership/mentorship portfolio

Secondary Users

4. College Administration

- **Roles:** Placement officers, department heads, deans, alumni relations team
- **Needs:** Improve placement rates, alumni engagement, accreditation metrics
- **Goals:**

- Track mentorship effectiveness
- Showcase alumni network strength
- Improve placement statistics
- Maintain quality of interactions

5. Potential Recruiters (Future Scope)

- **Description:** HR professionals from companies with multiple alumni
- **Needs:** Access to pre-vetted talent pipeline
- **Goals:** Identify candidates recommended by their own employees

4.2 User Prioritization

MVP Focus (Phase 1): Junior Students + Recent Alumni (0-3 years)

Phase 2 Expansion: Experienced alumni, admin analytics

Phase 3: Recruiter access, inter-college network

---

5. User Personas

Persona 1: Eager Junior Student

Name: Rohan Shah
Age: 20
Role: 3rd Year, Computer Engineering Student
Location: Ahmedabad, Gujarat

Background:

- CGPA: 8.5/10
- Skills: Python, basic Django, 100+ LeetCode problems solved
- Target Company: Odoo (heard it's great for Python developers)
- Career Goal: Backend SDE role at product company

Frustrations:

- "I don't know anyone who works at Odoo from my college"
- "LinkedIn cold messages never get replies"
- "YouTube videos give generic advice, not company-specific tips"
- "I'm not sure if I'm preparing the right things for Odoo interviews"

- "I want a referral but I don't have any connection"

Goals:

- Find alumni working at Odoo
- Understand Odoo's exact hiring process and tech stack
- Get a roadmap to prepare in next 6 months
- Build projects that Odoo values
- Eventually get a referral

Behavioral Traits:

- Proactive and motivated
- Willing to follow structured guidance
- Prefers written communication over calls
- Active on technical forums and GitHub
- Values actionable, specific advice

How ASCEND Helps:

- Discovers 8 alumni working at Odoo
- Asks structured question about preparation roadmap
- Receives detailed response from Odoo SDE-2 in 2 days
- Follows suggested projects and timeline
- Builds portfolio, then requests referral after 6 months

Persona 2: Busy Alumni Mentor

Name: Rahul Patel
Age: 25
Role: Senior Software Developer at Odoo
Location: Gandhinagar, India
Batch: 2020, Computer Engineering

Background:

- 3 years experience at Odoo
- Works with Python, PostgreSQL, Odoo framework

- Got his job through campus placement
- Active on LinkedIn with 5,000+ followers

Frustrations:

- "I get 10-15 LinkedIn messages per week asking 'how to get job at Odoo?'"
- "Students ask the same questions repeatedly"
- "I don't have time for 30-minute phone calls with every student"
- "Some students disappear after I spend time writing detailed advice"
- "No way to track if my advice actually helped anyone"

Goals:

- Help juniors from his college (wants to give back)
- Manage mentorship efficiently (limited to 2-3 hours/month)
- Give meaningful advice, not generic responses
- Build reputation as helpful mentor
- Potentially identify talented candidates for Odoo

Behavioral Traits:

- Prefers asynchronous communication (email/text over calls)
- Values well-researched, specific questions
- Willing to help serious, prepared students
- Appreciates when students follow through on advice
- Limited time but high quality contributions

How ASCEND Helps:

- Sets limit: max 10 questions per month
- Receives only structured questions with context
- Responds at his convenience (no real-time pressure)
- Sees impact through outcome feedback ("Your roadmap helped me get interview!")
- Builds trust score (94/100) showcasing his expertise
- Filters out low-effort questions automatically

Persona 3: Administrator/Placement Officer

Name: Dr. Neha Mehta
Age: 38
Role: Training & Placement Officer
Institution: College Engineering Department

Background:

- 15 years in academia
- Responsible for placement rates and industry relations
- Manages alumni database (3,000+ contacts)
- Organizes career guidance sessions

Frustrations:

- "Students don't know which alumni to contact for guidance"
- "Alumni engagement is low - most don't respond to college emails"
- "We organize talks but attendance is poor and impact is unclear"
- "No way to track which students got help from which alumni"
- "Accreditation requires proof of industry mentorship, but we only have anecdotal evidence"

Goals:

- Improve placement percentage from 65% to 80%
- Engage alumni network productively
- Provide structured mentorship to all students
- Track and report mentorship metrics for accreditation
- Build institutional knowledge repository

How ASCEND Helps:

- Central platform for all alumni-student interactions
- Dashboard showing engagement metrics (questions answered, response rates)
- Success stories with data (e.g., "10 students placed via referrals")
- Verified alumni profiles (LinkedIn-backed authentication)
- Moderation tools to maintain quality

- Reports for accreditation documentation

---

6. Use Cases and User Stories

6.1 Core Use Cases

Use Case 1: Student Discovers Company-Specific Mentors

Actor: Junior Student (Rohan)

Precondition: Rohan is registered and logged in

Trigger: Rohan wants to work at Odoo but doesn't know anyone there

Flow:

1. Rohan navigates to Company Directory
2. Searches for "Odoo" in search bar
3. Clicks on Odoo company card
4. Views company page showing 8 alumni from his college
5. Sees 3 mentors are currently accepting questions
6. Filters by "Currently Working" and "Interview Preparation"
7. Views Rahul Patel's profile (SDE-2, Trust Score 94/100)
8. Clicks "Ask Question" button

Postcondition: Rohan is redirected to structured question submission form with Odoo and Rahul pre-selected

Alternative Flow:

- If no alumni work at searched company → System shows "No alumni found at this company. Add to wishlist to be notified when someone joins."

Use Case 2: Student Submits Structured Question

Actor: Junior Student (Rohan)

Precondition: Student is on question submission form

Flow:

1. System displays form with Odoo and Rahul pre-filled
2. Rohan selects question type: "Career Roadmap"

3. System auto-fills his background (Year: 3rd, Branch: CS, Skills: Python, Django)
4. Rohan provides context:
    - Current status: "Completed 100 LC, built 2 Django projects"
    - Target: "SDE role at Odoo"
    - Timeline: "Want to apply in 6 months"
5. Rohan writes detailed question (500 words):
    - What should I learn/build specifically for Odoo?
    - Is my current preparation enough?
    - Which projects would impress Odoo recruiters?
6. AI system checks question quality
7. System shows suggestion: "Similar question answered 2 months ago - View Answer"
8. Rohan reviews but his question is more specific, proceeds
9. Rohan attaches resume (PDF)
10. Reviews preview and submits

Postcondition:

- Question enters Rahul's queue (position #3)
- Rohan receives confirmation email
- Rahul gets notification: "New question assigned"
- Question status: "Waiting for response"

Business Rules:

- Student can ask max 1 question per company per week
- Max 4 unanswered questions at a time
- Question must be 100-2000 characters
- AI flags low-quality questions for manual review

Use Case 3: Mentor Responds to Question

Actor: Alumni Mentor (Rahul)

Precondition: Question is in Rahul's queue

Flow:

1. Rahul logs in and sees notification badge (3 pending questions)
2. Navigates to "My Queue" dashboard
3. Sees questions sorted by FIFO order
4. Clicks on Rohan's question (submitted 1 day ago)
5. Reads full question with context:
    - Student background
    - Skills: Python, Django, PostgreSQL
    - Question: Roadmap for Odoo preparation
6. Clicks "Answer Question"
7. Uses rich text editor to write comprehensive response (800 words):
    - Month-by-month preparation plan
    - Specific skills needed (Python OOP, PostgreSQL advanced queries)
    - Project suggestions (build ERP module)
    - Interview process insights
8. Attaches resource link (Odoo's technical documentation)
9. Reviews response and submits
10. Marks question as "Answered"

Postcondition:

- Question moves from "In Progress" to "Answered"
- Rohan receives email notification with response
- Question removed from Rahul's active queue (now 2 pending)
- System schedules feedback request to Rohan (4 weeks later)

Alternative Flow:

- If Rahul needs more info → Clicks "Request Clarification" → Rohan gets notification to provide details
- If question is duplicate → Rahul can link to existing answer

---

Use Case 4: Student Provides Outcome Feedback

Actor: Junior Student (Rohan)

Precondition: 4 weeks have passed since receiving response

Flow:

1. Rohan receives email: "How was Rahul's advice? Provide feedback"
2. Clicks link, redirected to feedback form
3. System shows original question and Rahul's response
4. Rohan answers:
    - Was advice relevant? **Yes**
    - Did you implement it? **Yes, partially**
    - What was the result? **Improved skills - started building Odoo module project**
    - Rating: ⭐ ⭐ ⭐ ⭐ (4/5)
    - Comment: "Very detailed roadmap. Following month 2 now. Already feel more confident."
5. Submits feedback

Postcondition:

- Rahul's trust score increases: 94 → 96/100
- Rahul's statistics updated: +1 helpful response
- Feedback visible on Rahul's profile (anonymized)
- Rohan can mark question as "Helpful" in knowledge base

Business Rules:

- Feedback requested 4-6 weeks after response
- Optional but incentivized (gamification: "Complete feedback to unlock advanced features")
- Outcome-based, not just satisfaction rating

Use Case 5: Student Requests Referral

Actor: Junior Student (Rohan)

Precondition:

- Rohan has interacted with Rahul (asked 2 questions, both answered)
- Rohan has completed preparation (6 months passed)
- Rohan has built portfolio

Flow:

1. Rohan navigates to Rahul's profile
2. Clicks "Request Referral" button
3. System checks eligibility:
    - ✓ Has received guidance from Rahul
    - ✓ Trust score with Rahul > 70/100
    - ✓ Profile completeness > 80%
4. System shows referral request form
5. Rohan fills:
    - Target Role: "Junior Python Developer at Odoo"
    - Why you're ready:
        - "Completed roadmap you suggested 6 months ago"
        - "Built 2 Odoo modules (Inventory + CRM)"
        - "Solved 150+ LeetCode problems"
        - "Attached portfolio and project links"
    - Resume: (uploads latest version)
    - Portfolio links: GitHub, Live Demo
6. Rohan submits request

Postcondition:

- Rahul receives notification: "Referral request from Rohan"
- Request status: "Pending Review"
- Rohan sees: "Request sent. Rahul will review within 5 days."

Rahul's Review Flow:

1. Rahul views request with Rohan's portfolio

2. Clicks on GitHub links, reviews projects
3. Checks Rohan's progress over 6 months
4. Decision options:
    - **Approve:** "Will refer you. Here's the process..."
    - **Request Changes:** "Build one more project showing X skill"
    - **Decline:** "Not quite ready yet. Focus on Y first."
5. Rahul approves and provides next steps:
    - "Apply to Odoo careers page with this job ID"
    - "Mention referral code: RAHUL-REF-2026"
    - "I'll submit internal referral form"

Postcondition:

- Rohan receives approval notification
- Success story candidate (tracked for placement outcome)
- If placed → Rohan can submit success story

6.2 User Stories (Agile Format)

Epic 1: User Management

US-1.1: As a student, I want to register with my college email so that I can access the mentorship platform.
Acceptance Criteria:

- Registration form accepts college email domain only
- Email verification required before login
- Profile creation wizard after first login
- Account activated within 2 minutes

US-1.2: As an alumni, I want to register and get verified so that I can mentor students from my college.
Acceptance Criteria:

- Upload verification documents (degree/ID)
- LinkedIn profile verification

- Admin approval required
- Notification on approval/rejection

US-1.3: As a user, I want to login securely so that I can access my personalized dashboard.
Acceptance Criteria:

- Role-based redirection (student/alumni/admin)
- "Remember me" option
- Account lockout after 5 failed attempts
- Password reset functionality

Epic 2: Company Discovery

US-2.1: As a student, I want to browse companies where alumni work so that I can find mentors at my target companies.
Acceptance Criteria:

- Searchable company directory
- Filter by industry, size, location
- Company cards show alumni count and active mentors
- Click company → View detailed page

US-2.2: As a student, I want to view company-specific information so that I understand hiring process and culture.
Acceptance Criteria:

- Company page shows interview process
- Salary ranges (if available)
- Tech stack used
- Alumni success stories

US-2.3: As a student, I want to filter alumni by role and availability so that I find the most relevant mentor.
Acceptance Criteria:

- Filter: Current employees vs past
- Filter: By role (SDE, Tech Lead, etc.)
- Filter: By availability status

- Sort by trust score, response time

Epic 3: Question & Answer System

US-3.1: As a student, I want to submit structured questions so that mentors have context to provide relevant advice.
Acceptance Criteria:

- Form includes question type selection
- Auto-fill background from profile
- Mandatory context fields (current status, goals, specific question)
- AI quality checker flags low-quality questions
- Character limit: 100-2000 words

US-3.2: As a mentor, I want to see questions in a fair queue so that I can respond in order without being overwhelmed.
Acceptance Criteria:

- FIFO queue display
- Urgent questions highlighted (priority queue)
- Queue shows position number
- Can set max queue size
- Auto-pause when queue full

US-3.3: As a mentor, I want to respond with rich formatting so that I can provide clear, detailed guidance.
Acceptance Criteria:

- Rich text editor (bold, italic, lists, code blocks)
- Attach links and resources
- Preview before submit
- Save draft option
- Response logged with timestamp

US-3.4: As a student, I want to provide feedback on advice so that mentor quality is maintained.
Acceptance Criteria:

- Feedback requested 4 weeks after response
- Outcome-based questions (Did it work? What happened?)
- Rating scale + comments
- Impacts mentor trust score
- Feedback visible on mentor profile (anonymized)

Epic 4: Mentor Matching

US-4.1: As a student, I want the system to suggest best mentors so that I get matched with most relevant experts.
Acceptance Criteria:

- Algorithm considers: company match, skill overlap, availability, trust score
- Shows top 3 mentor suggestions
- Can manually select from all available mentors
- Auto-assign if no selection in 24 hours

US-4.2: As a mentor, I want to control my availability so that I don't get overwhelmed.
Acceptance Criteria:

- Set max questions per month
- Toggle "Accepting questions" on/off
- Set expected response time
- Pause/resume anytime

Epic 5: Referral System

US-5.1: As a student, I want to request referrals formally so that I can ask professionally after building credibility.
Acceptance Criteria:

- Eligibility check (interacted with mentor, profile complete)
- Referral request form with portfolio links
- Mentor can approve/decline/request changes
- Track referral status

US-5.2: As a mentor, I want to review referral requests so that I only refer qualified candidates.
Acceptance Criteria:

- View student's full portfolio
- See interaction history
- Options: Approve, Request changes, Decline with reason
- Provide next steps if approved

Epic 6: Knowledge Base

US-6.1: As a student, I want to search previously answered questions so that I can find information without asking duplicates.
Acceptance Criteria:

- Full-text search across all answered questions
- Filter by company, topic, date
- Sort by relevance, helpfulness
- "Was this helpful?" voting system

US-6.2: As a user, I want to browse common questions by company so that I can learn from others' experiences.
Acceptance Criteria:

- Company page shows top 10 common questions
- Click → View full Q&A thread
- Related questions suggested
- Can bookmark for later

Epic 7: Analytics & Admin

US-7.1: As an admin, I want to approve alumni registrations so that only verified users can mentor.
Acceptance Criteria:

- View pending registrations
- Verify LinkedIn profile

- Approve/reject with reason
- Send notification on decision

US-7.2: As an admin, I want to moderate questions so that quality is maintained.
Acceptance Criteria:

- View flagged questions
- Delete spam/inappropriate content
- Rollback moderation (undo using stack)
- Ban repeat offenders

US-7.3: As an admin, I want to view platform analytics so that I can track success metrics.
Acceptance Criteria:

- Dashboard showing: total users, questions answered, avg response time
- Charts: Engagement over time, top companies, mentor leaderboard
- Export reports for accreditation
- Success stories count

---

7. Functional Requirements

7.1 User Authentication & Authorization

FR-1.1: User Registration

- **Description:** System shall allow three types of users to register: Students, Alumni, and Admins
- **Priority:** P0 (Critical)
- **Inputs:** Name, Email, Password, Role-specific fields
- **Processing:**
    - Validate email domain (college-specific)
    - Hash password using bcrypt
    - Generate email verification token
    - Send verification email
- **Outputs:** User account created, verification email sent
- **Business Rules:**

    - Students: Must use college email (@ddu.ac.in)
    - Alumni: Can use personal email, requires admin verification
    - Password: Min 8 chars, 1 uppercase, 1 number, 1 special char
- **Error Handling:**
    - Duplicate email → Show "Email already registered"
    - Invalid domain → Show "Use college email"
    - Weak password → Show specific requirements

FR-1.2: Email Verification

- **Description:** System shall verify user email before activation
- **Priority:** P0 (Critical)
- **Processing:**
    - User clicks link in verification email
    - System validates token (24-hour expiry)
    - Activate account if valid
- **Outputs:** Account activated, redirect to login
- **Business Rules:**
    - Token expires after 24 hours
    - Can resend verification email (max 3 times)

FR-1.3: Role-Based Access Control

- **Description:** System shall enforce different permissions for Students, Alumni, and Admins
- **Priority:** P0 (Critical)
- **Access Matrix:**

Feature	Student	Alumni	Admin
Submit Questions	✓	✗	✗
Answer Questions	✗	✓	✗
View Company Directory	✓	✓	✓

Feature	Student	Alumni	Admin
Request Referrals	✓	✗	✗
Moderate Content	✗	✗	✓
View Analytics	✗	Limited	✓
Approve Users	✗	✗	✓

7.2 Profile Management

FR-2.1: Student Profile Creation

- **Description:** Students shall create comprehensive profiles including skills, projects, and career goals
- **Priority:** P0 (Critical)
- **Required Fields:**
  - Name, Batch Year, Branch, Roll Number
  - Skills (min 3 tags)
  - Career interests
  - Target companies (max 5)
- **Optional Fields:**
  - CGPA, Projects, Certifications, Social links
- **Processing:**
  - Calculate profile completeness score (0-100%)
  - Suggest improvements if <70%
- **Business Rules:**
  - Profile must be >50% complete to submit first question
  - Can update anytime

FR-2.2: Alumni Profile & Work History

- **Description:** Alumni shall maintain professional profiles with current and past employment
- **Priority:** P0 (Critical)

- **Required Fields:**
    - Current company, role, location
    - At least 1 work history entry
    - Expertise tags
    - Availability settings
- **Processing:**
    - Build company-alumni mapping (HashMap for O(1) lookup)
    - Calculate tenure at each company
    - Update

company statistics (alumni count)

- **Business Rules:**
    - Must mark one company as "current"
    - Can add unlimited past companies
    - Work history visible to all students

FR-2.3: Profile Privacy Settings

- **Description:** Users shall control visibility of certain profile fields
- **Priority:** P1 (Important)
- **Options:**
    - Students: Hide CGPA from mentors (Yes/No)
    - Alumni: Show/hide email address
    - Both: Profile visibility in directory
- **Default:** All fields visible except email

7.3 Company Directory & Discovery

FR-3.1: Company Master Data

- **Description:** System shall maintain database of companies where alumni work
- **Priority:** P0 (Critical)
- **Data Fields:**

    - Name, Logo, Website, Industry, Company Type
    - Headquarters, Employee count range
    - Is actively hiring (Yes/No)
- **Processing:**
    - Auto-create company when alumni adds work history
    - Admin can edit/merge duplicate entries
    - Calculate statistics: alumni count, avg trust score

FR-3.2: Company Search & Filtering

- **Description:** Students shall search and filter companies to find target companies
- **Priority:** P0 (Critical)
- **Search:** Full-text search on company name
- **Filters:**
    - Industry (IT Services, Product, Consulting, Finance)
    - Company Type (Startup, Mid-size, Large, MNC)
    - Location (City/Country)
    - Actively Hiring (checkbox)
- **Sort Options:**
    - Most alumni (default)
    - Alphabetical (A-Z)
    - Most active mentors
    - Fastest response time
- **Outputs:** Paginated list of company cards (20 per page)

FR-3.3: Company Detail Page

- **Description:** System shall display comprehensive company information
- **Priority:** P0 (Critical)
- **Sections:**
    - Header: Logo, name, industry, basic stats
    - Alumni List: Current and past employees with filters

- Company Intelligence: Hiring info, culture, tech stack (if available)
- Common Questions: Top 10 answered questions about this company
- Success Stories: Student testimonials

- **Interactions:**
  - Click alumni → View profile
  - Click "Ask Question" → Pre-fill company in form
  - Bookmark company → Add to student's wishlist

7.4 Question Submission System

FR-4.1: Structured Question Form

- **Description:** Students shall submit questions using mandatory structured format
- **Priority:** P0 (Critical)
- **Form Fields:**
  - Target Company (dropdown, searchable)
  - Preferred Mentor (optional, shows top 3 suggestions)
  - Question Type (dropdown): Interview Prep, Roadmap, Culture, Salary, Resume Review, Referral, General
  - Background Context (auto-filled from profile, editable)
  - Current Status (text, 100-500 chars): What have you done so far?
  - Specific Question (text, 500-2000 chars): What exactly do you need help with?
  - Urgency Level (Normal, High, Urgent)
  - Attachments (optional): Resume PDF, project links
- **Processing:**
  - AI quality checker analyzes question
  - Check for duplicates in knowledge base
  - Suggest similar answered questions
  - Validate all mandatory fields
- **Outputs:** Question submitted to mentor's queue

- **Business Rules:**
    - Student can ask max 1 question per company per week
    - Max 4 unanswered questions at a time
    - Cannot submit if profile <50% complete
    - High/Urgent questions require justification

FR-4.2: Duplicate Detection

- **Description:** System shall detect and flag potential duplicate questions
- **Priority:** P1 (Important)
- **Processing:**
    - Use TF-IDF or cosine similarity on question text
    - Compare against answered questions for same company
    - If similarity >70% → Show "Similar question found"
- **Outputs:**
    - List of similar questions with links
    - Option: "My question is different" to proceed
- **Business Rules:**
    - Doesn't block submission, only suggests
    - Admin can merge duplicates later

FR-4.3: AI Quality Checker

- **Description:** System shall analyze question quality before submission
- **Priority:** P1 (Important)
- **Checks:**
    - Length: 100-2000 characters
    - Specificity: Contains specific company/role/tech keywords
    - Clarity: Not too vague ("plz help", "urgent")
    - Context: Has sufficient background information
- **Outputs:**
    - Green: "Good question quality"

    - Yellow: "Consider adding more context"
    - Red: "Question is too vague, please revise"
- **Business Rules:**
    - Red flag → Requires manual admin approval
    - Yellow → Can submit but warned
    - Green → Auto-approved

7.5 Question Queue Management (DSA: FIFO Queue)

FR-5.1: Mentor Question Queue

- **Description:** Each mentor shall have a FIFO queue for fair question distribution
- **Priority:** P0 (Critical - Academic Requirement)
- **Data Structure:** Queue (First-In-First-Out)
- **Implementation:**
    - Use Python collections.deque for queue operations
    - Each mentor has individual queue instance
    - Max queue size configurable by mentor (default: 10)
- **Operations:**
    - enqueue(question): Add question to end of queue
    - dequeue(): Mentor picks next question (from front)
    - peek(): View next question without removing
    - size(): Get current queue length
    - is_full(): Check if queue at capacity
- **Business Logic:**
    - Questions answered in order received
    - Cannot skip questions (enforces fairness)
    - If queue full → Redirect to next available mentor
- **UI Display:**
    - Mentor sees queue position for each question

- Student sees "Position #3 in queue, estimated wait: 6 days"

FR-5.2: Priority Queue for Urgent Questions

- **Description:** Urgent questions shall be processed with higher priority
- **Priority:** P1 (Important - Academic Requirement)
- **Data Structure:** Min-Heap Priority Queue
- **Implementation:**
  - Use Python heapq module
  - Priority levels: 1 (Urgent <2 weeks), 2 (High 2-4 weeks), 3 (Normal)
- **Operations:**
  - push(priority, question): Add with priority value
  - pop(): Get highest priority question
- **Business Logic:**
  - Urgent questions jump to front of regular queue
  - Max 20% of questions can be urgent (prevent abuse)
  - Requires justification (interview date proof)
- **Processing:**
  - Check if student has legitimate urgency
  - Add to priority queue
  - Notify mentor immediately (email/SMS)

FR-5.3: Load Balancing

- **Description:** System shall distribute questions evenly among available mentors
- **Priority:** P1 (Important)
- **Algorithm:**
  1. Get all mentors at target company
  2. Filter by availability status (accepting questions)
  3. Sort by queue size (ascending)
  4. Assign to mentor with smallest queue
  5. If all full → Add to company waitlist

- **Business Rules:**
    - Student can manually select mentor (overrides auto-assignment)
    - If manual selection and mentor full → Waitlist for that specific mentor
    - Auto-assignment considers trust score as tiebreaker

7.6 Mentor Matching Algorithm (DSA: HashMap + Sorting)

FR-6.1: Company-Based Matching

- **Description:** System shall match students with most relevant mentors using scoring algorithm
- **Priority:** P0 (Critical - Academic Requirement)
- **Data Structure:** HashMap for company-to-mentors mapping
- **Implementation:**
    - Build HashMap: company_id → [list of mentor objects]
    - O(1) lookup for all mentors at a company
    - Python dictionary: {1: [mentor1, mentor2, ...], 2: [...]}
- **Processing:**

1. Lookup mentors at target company (O(1) via HashMap)
2. Calculate match score for each mentor
3. Sort mentors by score (Merge Sort O(n log n))
4. Return top 3 matches

- **Output:** Ranked list of suggested mentors

FR-6.2: Mentor Scoring Algorithm

- **Description:** System shall calculate compatibility score between student and mentor
- **Priority:** P0 (Critical)
- **Scoring Factors:**

Factor	Weight	Calculation
Currently at Company	50 points	+50 if current employee, +30 if past

Factor	Weight	Calculation
Years at Company	20 points	years × 5, max 20
Skill Overlap	20 points	Jaccard similarity × 20
Same Branch	10 points	+10 if match, +0 otherwise
Question Type Expertise	15 points	+15 if mentor's "can help with" includes this
Availability	-30 to +10	-30 if >90% full, +10 if <30% full
Trust Score	15 points	(trust_score/100) × 15
Response Rate	10 points	response_rate × 10
Recency	10 points	+10 current, +7 if <1yr, +4 if <2yr

- **Total Score Range:** 0-150 points
- **Algorithm:**

```
FOR each mentor at target_company:
  score = 0

  # Factor 1: Employment status
  IF mentor.current_company == target_company:
    score += 50
  ELSE IF mentor worked at target_company in past:
    score += 30

  # Factor 2: Experience at company
  tenure_years = calculate_tenure(mentor, target_company)
  score += min(tenure_years * 5, 20)

  # Factor 3: Skill overlap (Jaccard similarity)
  skill_similarity = len(student_skills ∩ mentor_skills) / len(student_skills ∪ mentor_skills)
  score += skill_similarity * 20
```

```
# Factor 4: Same academic background
IF student.branch == mentor.branch:
    score += 10

# Factor 5: Expertise in question type
IF question_type IN mentor.can_help_with:
    score += 15

# Factor 6: Availability (critical)
queue_utilization = mentor.queue_size / mentor.max_queue_size
IF queue_utilization > 0.9:
    score -= 30  # Heavy penalty
ELSE IF queue_utilization > 0.7:
    score -= 15  # Moderate penalty
ELSE IF queue_utilization < 0.3:
    score += 10  # Bonus for available mentors

# Factor 7: Historical performance
score += (mentor.trust_score / 100) * 15
score += mentor.response_rate * 10

# Factor 8: Recency of experience
months_since = months_since_worked_at(mentor, target_company)
IF months_since == 0:
    score += 10  # Currently working
ELSE IF months_since < 12:
    score += 7
```

```
  ELSE IF months_since < 24:
    score += 4

  mentor.match_score = score

SORT mentors BY match_score DESC
RETURN top 3 mentors
```

FR-6.3: Skill Similarity Calculation

- **Description:** System shall calculate skill overlap using Jaccard similarity
- **Priority:** P1 (Important)
- **Formula:** Similarity = |A ∩ B| / |A ∪ B|
    - A = Student's skills
    - B = Mentor's expertise
- **Example:**
    - Student skills: {Python, React, PostgreSQL, Django}
    - Mentor skills: {Python, PostgreSQL, JavaScript, Docker}
    - Intersection: {Python, PostgreSQL} = 2
    - Union: {Python, React, PostgreSQL, Django, JavaScript, Docker} = 6
    - Similarity: 2/6 = 0.33 → Score: 0.33 × 20 = 6.6 points

7.7 Response Management

FR-7.1: Mentor Response Interface

- **Description:** Mentors shall respond to questions with rich text formatting
- **Priority:** P0 (Critical)
- **Features:**
    - Rich text editor (bold, italic, underline, lists, code blocks, links)
    - Attach resources (URLs only, no file uploads in response)
    - Preview response before submit

    - Save draft (auto-save every 30 seconds)
    - Character limit: 200-5000 characters
- **Processing:**
    - Sanitize HTML to prevent XSS attacks
    - Parse and render Markdown
    - Store response with timestamp
    - Update question status: "Answered"
    - Remove from mentor's active queue
    - Notify student via email
- **Business Rules:**
    - Mentor can edit response within 24 hours
    - After 24 hours, edits require admin approval
    - One follow-up round allowed per question

FR-7.2: Follow-Up Questions

- **Description:** Students shall ask one follow-up question per thread
- **Priority:** P1 (Important)
- **Business Logic:**
    - After receiving response, student can click "Ask Follow-Up"
    - Follow-up limited to 500 characters
    - Must be related to original question
    - Mentor gets notification
    - Follow-up doesn't count toward weekly question limit
- **Business Rules:**
    - Max 1 follow-up per question
    - Follow-up must be submitted within 30 days of response
    - Mentor has 3 days to respond to follow-up

FR-7.3: Response Time Tracking

- **Description:** System shall track and display mentor response times

- **Priority:** P1 (Important)
- **Metrics Calculated:**
    - Individual question response time: answered_at - submitted_at
    - Mentor average response time: mean(all response times)
    - Company average response time: mean(all mentors at company)
- **Display:**
    - On mentor profile: "Avg response: 1.8 days"
    - On company page: "Avg response: 2.3 days"
    - On question: "Responded in 2 days"
- **Business Rules:**
    - Only count business days (exclude weekends)
    - Response time >7 days flagged as "slow"
    - Impacts mentor trust score if consistently slow

7.8 Trust Score System

FR-8.1: Outcome-Based Feedback

- **Description:** Students shall provide feedback on advice effectiveness after implementation period
- **Priority:** P0 (Critical)
- **Timing:** Feedback requested 4-6 weeks after response received
- **Feedback Form:**
    - Was the advice relevant to your question? (Yes/No)
    - Did you implement the advice? (Yes/Partially/No/Too early to tell)
    - What was the outcome?
        - Got job/internship offer (⭐ ⭐ ⭐ ⭐ ⭐)
        - Interview calls increased (⭐ ⭐ ⭐ ⭐)
        - Skills improved significantly (⭐ ⭐ ⭐)
        - Somewhat helpful (⭐ ⭐)

        - Not helpful (⭐)
    - Additional comments (optional, 500 chars max)
- **Processing:**
    - Calculate points based on outcome
    - Update mentor's trust score
    - Add to mentor's statistics
- **Business Rules:**
    - Feedback is optional but incentivized
    - Late feedback (>90 days) has reduced weight
    - Can update feedback if outcome changes

FR-8.2: Trust Score Calculation

- **Description:** System shall calculate mentor trust score based on feedback
- **Priority:** P0 (Critical)
- **Formula:**

Trust Score = Weighted Average of Outcomes

Outcome Points:

- Got job/internship: +10 points

- Interview calls increased: +7 points

- Skills improved: +5 points

- Advice was relevant: +2 points

- Somewhat helpful: +1 point

- Not helpful: -3 points

- No response given (unanswered): -10 points

Trust Score = (Total Points / Total Responses) * 10

Normalized to 0-100 scale

Initial Trust Score: 50/100 (neutral, for new mentors)

- **Additional Factors:**
    - Response rate: (answered / assigned) × 20% bonus
    - Timeliness: Average response time <3 days → +5 points
    - Consistency: >10 responses without negative feedback → +5 points
- **Display:**
    - Mentor profile: "Trust Score: 94/100 ⭐ ⭐ ⭐ ⭐ ⭐ "
    - Color coding:
        - 90-100: Dark Green (Excellent)
        - 75-89: Green (Very Good)
        - 60-74: Yellow (Good)
        - 40-59: Orange (Average)
        - <40: Red (Poor)

FR-8.3: Trust Score Decay

- **Description:** Trust scores shall decay over time for inactive mentors
- **Priority:** P2 (Nice to have)
- **Logic:**
    - If mentor hasn't answered questions in 6 months → Gradual score reduction (-1 point/month)
    - If mentor goes inactive (turns off availability) → Score frozen (no decay)
    - Purpose: Keep scores relevant to current engagement
- **Business Rules:**
    - Minimum score after decay: 40/100
    - Reactivation resets decay

7.9 Referral System

FR-9.1: Referral Eligibility Check

- **Description:** System shall verify student eligibility before allowing referral request

- **Priority:** P1 (Important)
- **Eligibility Criteria:**
    - Has interacted with mentor (at least 1 question answered)
    - Trust level with mentor >70/100 (based on feedback given)
    - Profile completeness >80%
    - Has portfolio/projects showcased
    - No pending unanswered questions
- **Processing:**
    - Check all criteria when student clicks "Request Referral"
    - If not eligible → Show specific missing requirements
    - If eligible → Show referral request form
- **Business Rules:**
    - Can request referral from mentor even if they're not current employee (if worked there recently)
    - Max 1 active referral request per mentor at a time
    - Can request from multiple mentors for different companies

FR-9.2: Referral Request Submission

- **Description:** Students shall submit formal referral requests with portfolio
- **Priority:** P1 (Important)
- **Request Form:**
    - Target Role (text, e.g., "Junior Python Developer")
    - Why you're ready (text, 500-1000 chars):
        - Preparation completed
        - Projects built
        - Skills acquired
    - Resume Upload (PDF, max 2MB)
    - Portfolio Links (GitHub, LinkedIn, live projects)
    - Additional Notes (optional)

- **Processing:**
    - Validate all attachments
    - Create referral request record
    - Notify mentor
    - Set status: "Pending Review"
- **Outputs:**
    - Student: "Request sent. Mentor will review within 5 business days."
    - Mentor: Email + in-app notification

FR-9.3: Mentor Referral Review

- **Description:** Mentors shall review and respond to referral requests
- **Priority:** P1 (Important)
- **Review Interface:**
    - View student's full profile and portfolio
    - See interaction history (questions asked, responses given)
    - Review projects and resume
    - Decision options:
        - **Approve:** Agree to refer
        - **Request Changes:** Ask for specific improvements
        - **Decline:** Not ready yet, provide constructive feedback
- **Approval Flow:**
    - Mentor provides next steps:
        - "Apply to [Company] careers page with Job ID: XXXX"
        - "Mention referral code: MENTOR-NAME-REF-2026"
        - "I will submit internal referral form"
        - Expected timeline
    - System tracks referral status
    - If student gets interview → Update status to "Interview Scheduled"
    - If student gets offer → Update status to "Offer Received" → Success story

- **Business Rules:**
    - Mentor must respond within 7 days (else auto-declined)
    - Declined students can reapply after 3 months
    - Mentor can revoke referral if student misrepresents information

---

7.10 Knowledge Base

FR-10.1: Q&A Archive

- **Description:** All answered questions shall be searchable in knowledge base
- **Priority:** P1 (Important)
- **Indexing:**
    - Full-text search on question and answer text
    - Tag by company, question type, skills mentioned
    - Auto-generate keywords using TF-IDF
- **Search Features:**
    - Search by keywords
    - Filter by company, question type, date range
    - Sort by relevance, date, helpfulness
- **Privacy:**
    - Student names anonymized ("Student from 3rd Year CS")
    - Mentor names visible with consent
    - Sensitive information redacted (salary numbers, personal details)

FR-10.2: Helpful Voting System

- **Description:** Users shall vote on answer helpfulness
- **Priority:** P2 (Nice to have)
- **Actions:**
    - "Was this helpful?" Yes/No buttons
    - Vote count displayed: "47 students found this helpful"
    - Most helpful answers rise to top in search results

- **Business Rules:**
    - One vote per user per answer
    - Cannot vote on own questions/answers
    - Voting contributes to mentor's trust score

FR-10.3: Duplicate Question Linking

- **Description:** Admins and mentors shall link duplicate questions to canonical answers
- **Priority:** P2 (Nice to have)
- **Process:**
    - When answering, mentor can select "This is duplicate of Question #123"
    - System redirects future viewers to original answer
    - Student notified: "Similar question already answered, see here"
- **Benefits:**
    - Reduces mentor workload
    - Builds comprehensive knowledge base
    - Students find answers faster

7.11 Company Intelligence

FR-11.1: Alumni Contribution System

- **Description:** Alumni shall contribute insights about their companies
- **Priority:** P1 (Important)
- **Contribution Form:**
    - Hiring Information: Typical months, common roles, hiring duration
    - Compensation: Salary ranges (optional, can be anonymized)
    - Interview Process: Number of rounds, difficulty rating, topics covered
    - Culture: Work-life balance, learning opportunities, remote policy
    - Tech Stack: Languages, frameworks, tools used
    - Best For: Tags describing company strengths
- **Processing:**

    - Aggregate data from multiple alumni
    - Calculate averages for numerical fields
    - Combine unique items for lists (tech stack, skills required)
    - Admin reviews before publishing
- **Business Rules:**
    - Min 3 alumni contributions before displaying aggregate data
    - Salary data always shown as ranges, never exact
    - Contributors can update their inputs anytime

FR-11.2: Interview Process Mapping

- **Description:** System shall display typical interview process for each company
- **Priority:** P1 (Important)
- **Display Format:**

Interview Process for Odoo:

├── Round 1: Online Assessment (2 hours)

│ Topics: Python fundamentals, SQL queries, DSA (medium level)

│

├── Round 2: Technical Interview (1 hour)

│ Topics: System design (LLD), OOP concepts, Code review

│

├── Round 3: Project Discussion (45 mins)

│ Topics: Past projects, problem-solving approach

│

└── Round 4: HR & Culture Fit (30 mins)

Topics: Career goals, team fit, salary negotiation

Difficulty Rating: 7/10

Avg Duration: Apply → Offer in 4-6 weeks

Success Rate: 15-20% (based on alumni reports)

FR-11.3: Tech Stack Display

- **Description:** Show detailed technology stack used at each company
- **Priority:** P1 (Important)
- **Categorized Display:**
    - Languages: Python, JavaScript, etc.
    - Frameworks: Django, React, Vue.js
    - Databases: PostgreSQL, MongoDB, Redis
    - Tools & Platforms: Docker, AWS, Git, Jenkins
    - Domain Knowledge: ERP systems, Cloud Architecture
- **Use Case:** Students know exactly what to learn for target company

---

7.12 Admin Panel

FR-12.1: Alumni Verification

- **Description:** Admins shall verify and approve alumni registrations
- **Priority:** P0 (Critical)
- **Approval Workflow:**
    1. View pending registrations queue
    2. For each registration:
        - Review submitted documents
        - Verify LinkedIn profile (cross-check company)
        - Check if company exists in database
        - Validate batch year and branch
    3. Decision:
        - Approve → Send welcome email, activate account
        - Reject → Provide reason, send rejection email
        - Request More Info → Ask for additional documents
- **Verification Checklist:**
    - ✓ LinkedIn profile matches submitted information

    - ✓ Current company verifiable
    - ✓ Graduation year matches college records
    - ✓ Documents are authentic (not photoshopped)
- **SLA:** Approve/reject within 48 hours

FR-12.2: Content Moderation

- **Description:** Admins shall moderate questions and answers for quality
- **Priority:** P0 (Critical)
- **Moderation Queue:**
    - AI-flagged questions (too vague, spam, inappropriate)
    - Reported content (users can report violations)
    - First-time user questions (manual review)
- **Actions Available:**
    - Approve → Assign to mentor
    - Edit → Fix grammar, remove sensitive info, then approve
    - Request Revision → Send back to student with feedback
    - Delete → Remove completely with reason
- **Moderation Stack (DSA: LIFO):**
    - Admins can "undo" last moderation action
    - Stack stores last 10 actions
    - Useful for accidental deletions
    - Implementation: Python list as stack

FR-12.3: Analytics Dashboard

- **Description:** Admins shall view comprehensive platform analytics
- **Priority:** P1 (Important)
- **Metrics Displayed:**
    - **User Stats:** Total students, alumni, active users, new registrations (this month)
    - **Engagement:** Questions asked/answered, avg response time, response rate %
    - **Company Stats:** Top 10 companies by student interest, alumni distribution

    - **Mentor Performance:** Leaderboard by trust score, most active mentors
    - **Success Metrics:** Referrals requested/approved, placements tracked, success stories
    - **Time-Series Charts:** Questions over time, response time trends, user growth
- **Export Options:**
    - Download reports as PDF/Excel
    - Filters: Date range, company, user type
- **Use Case:** Reports for college accreditation, annual reviews, stakeholder presentations

FR-12.4: User Management

- **Description:** Admins shall manage user accounts and permissions
- **Priority:** P1 (Important)
- **Actions:**
    - View all users (students, alumni, admins)
    - Search/filter users
    - Edit user profiles (correct errors)
    - Suspend/ban users (for policy violations)
    - Reset passwords (if user requests)
    - Promote student to alumni (after graduation)
    - Assign/revoke admin privileges
- **Business Rules:**
    - Cannot delete users (only suspend for data integrity)
    - All actions logged for audit trail
    - Banned users cannot create new accounts with same email

---

7.13 Notification System

FR-13.1: Email Notifications

- **Description:** System shall send email notifications for key events
- **Priority:** P1 (Important)

- **Notification Triggers:**

Event	Recipient	Email Content
New Question Assigned	Mentor	"New question from [Student] about [Company]" + Link
Question Answered	Student	"Your question was answered by [Mentor]" + Preview
Feedback Request	Student	"[Mentor] responded 4 weeks ago. How did it go?"
Referral Request	Mentor	"[Student] requested referral for [Role]" + Portfolio link
Referral Status Update	Student	"Your referral request was [Approved/Declined]" + Details
Alumni Application Approved	Alumni	"Welcome to ASCEND! Your mentor account is active."
Account Verification	All	"Verify your email address" + Verification link
Weekly Digest	Mentors	"You have [X] pending questions" + Summary

- **Email Features:**
    - HTML templates with branding
    - Unsubscribe link (optional notifications only)
    - CTA buttons linking back to platform
    - Mobile-responsive design
- **Business Rules:**
    - Critical notifications (account verification, referral updates) cannot be disabled
    - Optional notifications (weekly digest) can be toggled in settings
    - Rate limit: Max 5 emails per user per day (prevent spam)

FR-13.2: In-App Notifications

- **Description:** Users shall see real-time notifications within application
- **Priority:** P2 (Nice to have)
- **Notification Bell:**

    - Icon in navbar with badge count
    - Click → Dropdown showing recent notifications
    - Mark as read/unread
    - "View All" link to notifications page
- **Notification Types:**
    - Info: New features, system updates
    - Action Required: "Respond to question", "Provide feedback"
    - Success: "Your question was answered"
    - Warning: "Your profile is incomplete"
- **Persistence:**
    - Store in database
    - Show unread count on login
    - Auto-mark read after viewing

7.14 Search & Discovery

FR-14.1: Global Search

- **Description:** Users shall search across companies, alumni, and questions
- **Priority:** P1 (Important)
- **Search Scope:**
    - Companies (by name, industry)
    - Alumni (by name, company, skills)
    - Questions (by content, tags)
- **Search Features:**
    - Auto-suggest as user types
    - Fuzzy matching for typos
    - Filter results by type (Companies/People/Q&A)
    - Paginated results
- **Search Algorithm:**

    - Full-text search on indexed fields
    - Ranking by relevance (TF-IDF)
    - Boost exact matches

FR-14.2: Alumni Directory

- **Description:** Students shall browse all verified alumni
- **Priority:** P2 (Nice to have)
- **Directory Features:**
    - Grid/List view toggle
    - Filter by:
        - Current company
        - Batch year
        - Branch
        - Currently accepting questions (Yes/No)
        - Availability status
    - Sort by:
        - Name (A-Z)
        - Trust score (High to Low)
        - Most active
        - Recently joined
- **Alumni Card Display:**
    - Photo, Name, Batch Year
    - Current company & role
    - Trust score & questions answered
    - "View Profile" or "Ask Question" button

---

8. Non-Functional Requirements

8.1 Performance Requirements

NFR-1.1: Page Load Time

- **Requirement:** All pages shall load within 2 seconds on 4G connection
- **Measurement:** 95th percentile load time <2 seconds
- **Critical Pages:** Company directory, company detail page, student dashboard
- **Optimization Strategies:**
    - Database query optimization (indexes on foreign keys)
    - Pagination (max 20 items per page)
    - Image lazy loading
    - Caching frequently accessed data (company list, mentor profiles)
    - Minimize JavaScript bundle size

NFR-1.2: Database Query Performance

- **Requirement:** All database queries shall execute within 100ms
- **Critical Queries:**
    - Mentor matching algorithm: <500ms
    - Company search with filters: <200ms
    - User authentication: <50ms
- **Optimization:**
    - Indexes on: email, company_id, mentor_id, status
    - Denormalized fields for statistics (alumni_count, avg_trust_score)
    - Query result caching for 5 minutes

NFR-1.3: Concurrent Users

- **Requirement:** System shall support 500 concurrent users without degradation
- **Peak Load:** During placement season (August-December)
- **Load Testing:** Use Locust or JMeter to simulate load
- **Scalability Plan:**
    - Horizontal scaling

(multiple Flask instances behind load balancer)

- Database connection pooling
- Session management via Redis

NFR-1.4: API Response Time

- **Requirement:** REST API endpoints shall respond within 200ms (95th percentile)
- **Critical Endpoints:**
    - GET /api/companies/search
    - GET /api/mentors/match
    - POST /api/questions/submit
    - GET /api/questions/:id
- **Monitoring:** Log response times, alert if >500ms

8.2 Reliability & Availability

NFR-2.1: Uptime

- **Requirement:** System shall have 99% uptime (excluding planned maintenance)
- **Downtime Budget:** Max 7.2 hours/month
- **Planned Maintenance:** Sundays 2-4 AM IST (low traffic)
- **Monitoring:** Health check endpoint every 60 seconds

NFR-2.2: Data Backup

- **Requirement:** Database shall be backed up daily
- **Backup Strategy:**
    - Full backup: Daily at 3 AM IST
    - Incremental backup: Every 6 hours
    - Retention: 30 days
    - Storage: Cloud storage (AWS S3 or equivalent)
- **Recovery:**
    - Recovery Point Objective (RPO): 6 hours (max data loss)
    - Recovery Time Objective (RTO): 2 hours (max downtime)

NFR-2.3: Error Handling

- **Requirement:** System shall handle errors gracefully without data loss
- **Error Types:**

    - Client errors (4xx): Show user-friendly message
    - Server errors (5xx): Log error, show generic message, notify admin
    - Database errors: Retry logic (3 attempts), fallback to cached data
- **Logging:**
    - All errors logged with timestamp, user ID, stack trace
    - Critical errors trigger email alerts to admin

8.3 Security Requirements

NFR-3.1: Authentication Security

- **Requirement:** User passwords shall be hashed using industry-standard algorithms
- **Implementation:**
    - Algorithm: bcrypt with salt (cost factor: 12)
    - Password policy: Min 8 chars, 1 uppercase, 1 lowercase, 1 number, 1 special char
    - Session timeout: 24 hours (or on browser close if "Remember Me" not selected)
    - Account lockout: 5 failed attempts → 15-minute cooldown
- **Session Management:**
    - Use Flask-Login for session handling
    - CSRF tokens on all forms
    - Secure cookies (HttpOnly, Secure flags in production)

NFR-3.2: Data Protection

- **Requirement:** Sensitive user data shall be protected in transit and at rest
- **Encryption:**
    - HTTPS enforced in production (TLS 1.2+)
    - Database encryption at rest (if supported by hosting provider)
    - Email addresses hashed in logs
- **Privacy:**
    - Student CGPA hidden if privacy setting enabled

    - Alumni emails not exposed to students (communication via platform)
    - Personal data (phone, address) never collected

NFR-3.3: SQL Injection Prevention

- **Requirement:** All database queries shall use parameterized queries (ORM)
- **Implementation:**
    - Use SQLAlchemy ORM exclusively (no raw SQL)
    - Validate all user inputs
    - Sanitize HTML in question/answer text (prevent XSS)
- **Testing:** Automated security scans using OWASP ZAP or similar

NFR-3.4: Access Control

- **Requirement:** Users shall only access data they are authorized to view
- **Enforcement:**
    - Role-based access control (RBAC)
    - Students cannot view other students' unanswered questions
    - Alumni can only see questions assigned to them
    - Admins have full access with audit logging
- **Authorization Checks:**
    - Middleware validates user role on every request
    - API endpoints return 403 Forbidden if unauthorized

NFR-3.5: Rate Limiting

- **Requirement:** API endpoints shall have rate limits to prevent abuse
- **Limits:**
    - Login endpoint: 5 attempts per minute per IP
    - Question submission: 1 per hour per student
    - Search API: 100 requests per hour per user
    - Email sending: 5 emails per day per user
- **Implementation:** Use Flask-Limiter extension
- **Response:** HTTP 429 Too Many Requests with retry-after header

---

8.4 Usability Requirements

NFR-4.1: User Interface

- **Requirement:** UI shall be intuitive and require minimal training
- **Design Principles:**
    - Mobile-first responsive design (Bootstrap grid)
    - Consistent navigation (navbar present on all pages)
    - Clear Call-to-Action buttons (primary action always prominent)
    - Form validation with real-time feedback
    - Breadcrumbs for navigation context
- **Accessibility:**
    - WCAG 2.1 Level AA compliance (target)
    - Semantic HTML5 tags
    - Alt text for all images
    - Keyboard navigation support
    - Minimum contrast ratio 4.5:1

NFR-4.2: Browser Compatibility

- **Requirement:** Application shall work on modern browsers
- **Supported Browsers:**
    - Chrome 90+ (primary)
    - Firefox 88+
    - Safari 14+
    - Edge 90+
- **Not Supported:** IE 11 (show deprecation message)
- **Testing:** Cross-browser testing using BrowserStack or LambdaTest

NFR-4.3: Mobile Responsiveness

- **Requirement:** All features shall be accessible on mobile devices (viewport 360px+)
- **Responsive Breakpoints:**

    - Mobile: <768px
    - Tablet: 768px-1024px
    - Desktop: >1024px
- **Mobile-Specific Optimizations:**
    - Touch-friendly buttons (min 44px tap target)
    - Collapsible navigation menu
    - Optimized images for mobile bandwidth
    - No horizontal scrolling

NFR-4.4: Internationalization (Future Scope)

- **Requirement:** System architecture shall support multiple languages
- **Implementation:**
    - Separate UI text from code (use i18n library)
    - Database support for Unicode (UTF-8)
    - Date/time formatting based on locale
- **Phase 1:** English only
- **Phase 2:** Add Gujarati/Hindi

8.5 Maintainability Requirements

NFR-5.1: Code Quality

- **Requirement:** Code shall follow PEP 8 style guide (Python)
- **Tools:**
    - Linting: Pylint, Flake8
    - Formatting: Black (auto-formatter)
    - Type hints: MyPy (static type checking)
- **Code Reviews:**
    - All code reviewed before merge (peer review or CodeRabbit AI)
    - Min 80% test coverage required

NFR-5.2: Documentation

- **Requirement:** All modules and functions shall be documented
- **Documentation Types:**
    - Inline comments for complex logic
    - Docstrings for all functions (Google style)
    - README.md with setup instructions
    - API documentation (Swagger/OpenAPI)
    - User manual (for students, alumni, admins)
- **Maintenance:**
    - Documentation updated with every feature change
    - Architecture diagrams (ERD, system design)

NFR-5.3: Modularity

- **Requirement:** System shall be modular for easy feature additions
- **Architecture:**
    - Blueprint-based Flask app (separate modules for auth, companies, questions, admin)
    - Service layer for business logic
    - Reusable utility functions
    - Configuration separate from code (environment variables)
- **Benefits:**
    - Easy to add new features without breaking existing
    - Can disable features via config flags
    - Unit testing simplified

8.6 Scalability Requirements

NFR-6.1: User Growth

- **Requirement:** System shall scale to support 5,000 users within 2 years
- **Current Capacity:** 500 users (Year 1)
- **Scaling Strategy:**

    - Database: PostgreSQL supports millions of rows
    - Application: Stateless design allows horizontal scaling
    - Caching: Redis for session and frequently accessed data
    - CDN: Static assets served via CDN (images, CSS, JS)

NFR-6.2: Data Growth

- **Requirement:** System shall handle 10,000+ questions without performance degradation
- **Database Design:**
    - Proper indexing on search fields
    - Archiving old data (questions >2 years old moved to archive table)
    - Pagination on all list views (prevent loading all records)

NFR-6.3: Multi-College Support (Future)

- **Requirement:** Architecture shall support multiple colleges
- **Design Considerations:**
    - Add college_id foreign key to user tables
    - Data isolation per college
    - Shared company database across colleges
    - College-specific branding (logo, colors)
- **Phase 1:** Single college (DDU)
- **Phase 2:** Multi-tenant architecture

8.7 Compliance & Legal

NFR-7.1: Data Privacy

- **Requirement:** System shall comply with data privacy best practices
- **Implementation:**
    - Privacy Policy page (what data collected, how used)
    - Terms of Service (user responsibilities)
    - Cookie consent banner (if analytics added)

    - User can request data deletion (GDPR-inspired)
- **Data Retention:**
    - Active accounts: Indefinite
    - Deleted accounts: Personal data purged after 30 days
    - Anonymous usage data retained for analytics

NFR-7.2: Content Moderation

- **Requirement:** Platform shall prevent inappropriate content
- **Policies:**
    - Code of Conduct (respectful communication, no harassment)
    - Prohibited content: Hate speech, spam, plagiarism, false information
    - Reporting mechanism for users to flag violations
- **Enforcement:**
    - AI flagging of inappropriate keywords
    - Manual admin review
    - Warning system (1st offense: warning, 2nd: suspension, 3rd: ban)

NFR-7.3: Intellectual Property

- **Requirement:** User-generated content ownership shall be clear
- **Terms:**
    - Students/Alumni retain ownership of their content
    - Platform has license to display/distribute content within platform
    - Users grant permission for anonymized content in knowledge base
    - No commercial use of content without consent

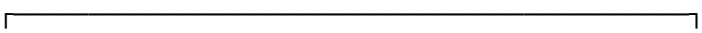
---

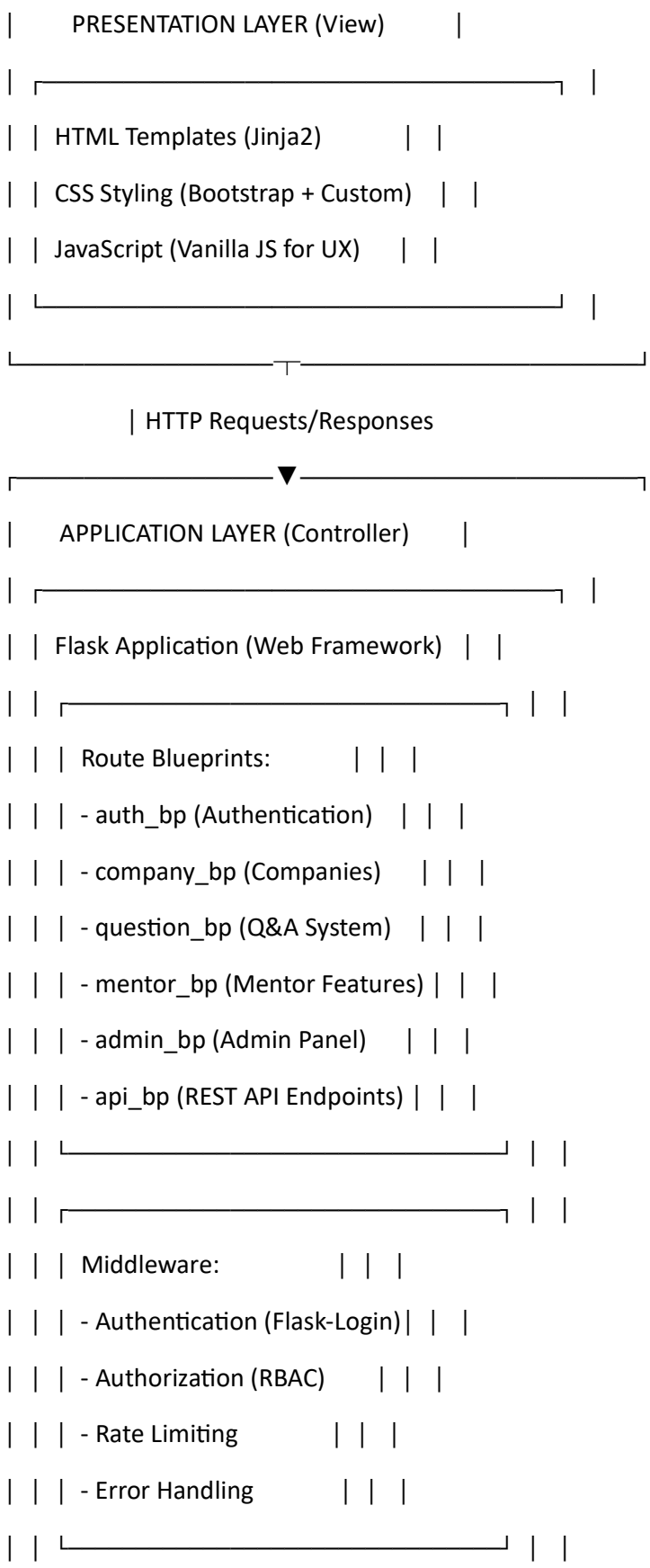
9. System Architecture

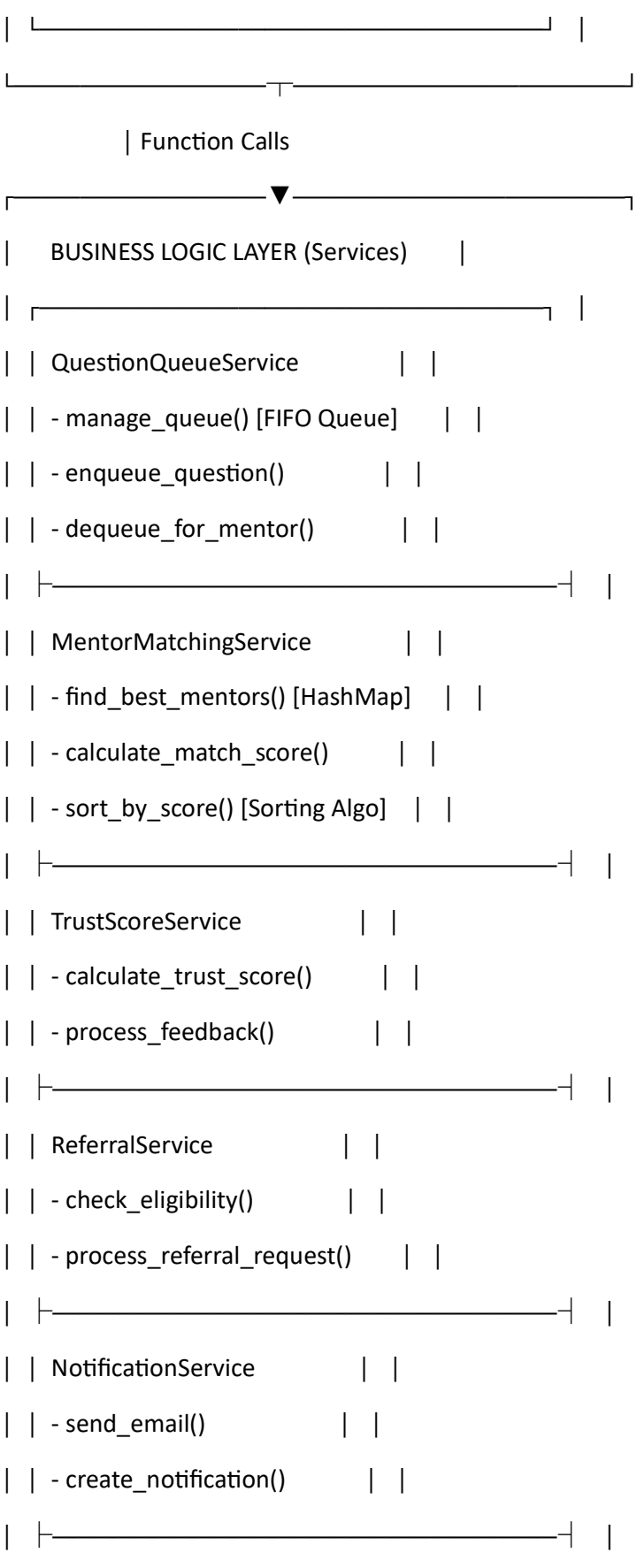
9.1 High-Level Architecture

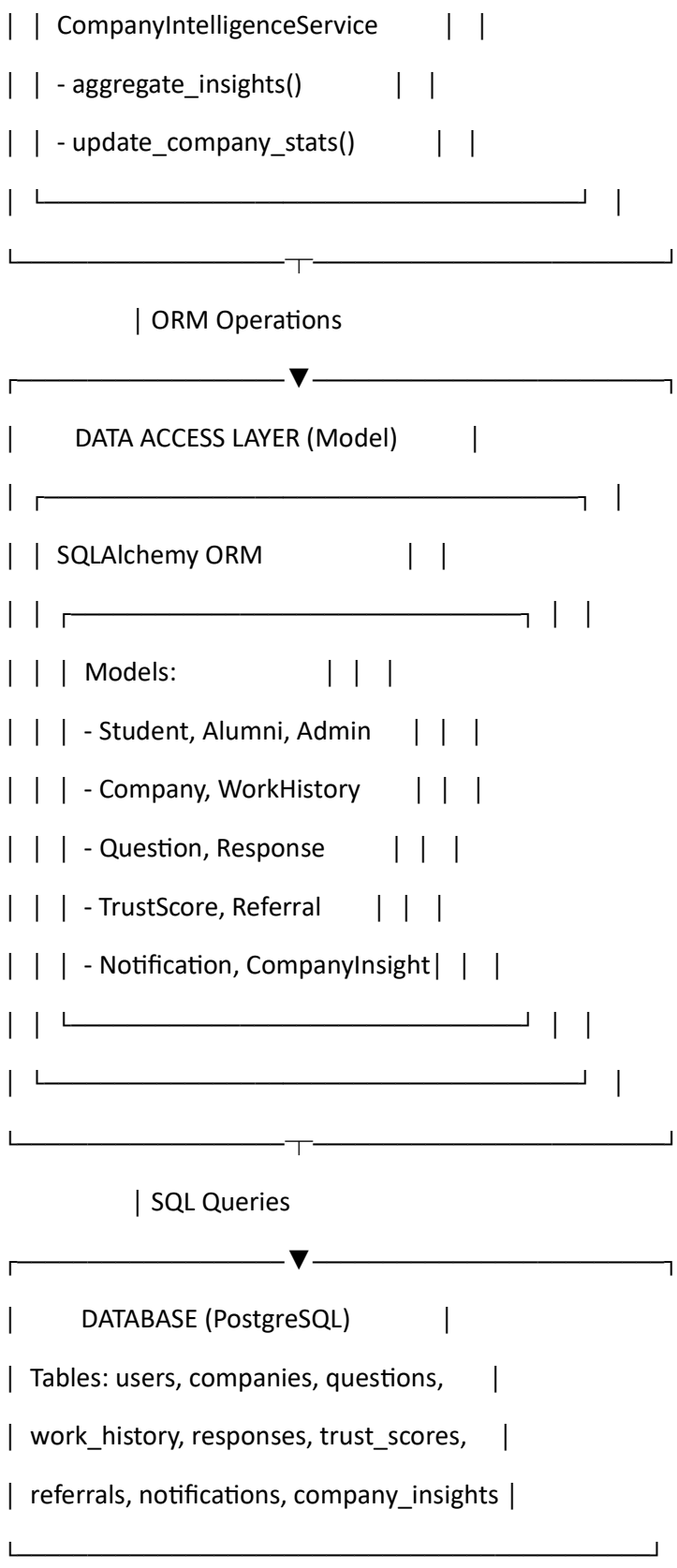
Architecture Pattern: Model-View-Controller (MVC) / Layered Architecture

System Components:




PRESENTATION LAYER (View)
HTML Templates (Jinja2)
CSS Styling (Bootstrap + Custom)
JavaScript (Vanilla JS for UX)
HTTP Requests/Responses
APPLICATION LAYER (Controller)
Flask Application (Web Framework)
Route Blueprints:
- auth_bp (Authentication)
- company_bp (Companies)
- question_bp (Q&A System)
- mentor_bp (Mentor Features)
- admin_bp (Admin Panel)
- api_bp (REST API Endpoints)
Middleware:
- Authentication (Flask-Login)
- Authorization (RBAC)
- Rate Limiting
- Error Handling


Function Calls
BUSINESS LOGIC LAYER (Services)
QuestionQueueService
- manage_queue() [FIFO Queue]
- enqueue_question()
- dequeue_for_mentor()
MentorMatchingService
- find_best_mentors() [HashMap]
- calculate_match_score()
- sort_by_score() [Sorting Algo]
TrustScoreService
- calculate_trust_score()
- process_feedback()
ReferralService
- check_eligibility()
- process_referral_request()
NotificationService
- send_email()
- create_notification()



9.2 Technology Stack Summary

Layer	Technology	Purpose
Frontend	HTML5	Structure
	CSS3 + Bootstrap 5	Styling & Responsive Design
	JavaScript (ES6+)	Client-side Interactivity
Backend	Python 3.10+	Programming Language
	Flask 2.3+	Web Framework
	Flask-Login	Authentication & Session Management
	Flask-Limiter	Rate Limiting
Database	PostgreSQL 14+	Relational Database
	SQLAlchemy 2.0+	ORM (Object-Relational Mapping)
Email	Flask-Mail	Email Notifications
Deployment	Gunicorn	WSGI HTTP Server
	Nginx	Reverse Proxy & Static Files
	Docker (optional)	Containerization
Hosting	Railway / Render / Heroku	Cloud Platform (development)
	AWS / GCP (future)	Production Hosting
Version Control	Git + GitHub	Code Repository

9.3 Data Structures & Algorithms Integration

Academic Requirement: Demonstrate practical application of DSA concepts

DSA Concept	Implementation Location	Purpose	Complexity
Queue (FIFO)	services/question_queue_service.py	Fair question distribution to mentors	Enqueue: O(1), Dequeue: O(1)
Priority Queue	services/urgent_queue_service.py	Handle urgent/deadline-based questions first	Insert: O(log n), Extract-Min: O(log n)

DSA Concept	Implementation Location	Purpose	Complexity
Stack (LIFO)	services/admin_moderation_service.py	Undo moderation actions (rollback)	Push: O(1), Pop: O(1)
HashMap	services/mentor_matching_service.py	Company-to-mentors mapping for O(1) lookup	Lookup: O(1), Insert: O(1)
Sorting (Merge Sort)	services/mentor_matching_service.py	Rank mentors by match score	O(n log n)
String Similarity	utils/duplicate_detection.py	Detect duplicate questions (TF-IDF/Cosine)	O(n×m) where n,m = string lengths
Graph (Future)	services/network_visualization.py	Alumni network visualization (Phase 2)	DFS/BFS: O(V+E)

DSA Implementation Example - Queue:

```
# Conceptual implementation (not actual code per PRD format)
# File: services/question_queue_service.py

Class: MentorQuestionQueue
  Properties:
    - queue: deque (collections.deque for O(1) operations)
    - max_size: integer (mentor-defined limit)
    - mentor_id: integer

  Methods:
    enqueue(question):
      - Check if queue.size() < max_size
      - If yes: queue.append(question)  # O(1)
      - If no: redirect_to_next_available_mentor()
```

dequeue():

- Return queue.popleft() # O(1), FIFO order

peek():

- Return queue[0] without removing

is_full():

- Return queue.size() >= max_size

---

10. Data Models

10.1 Entity Relationship Diagram (Conceptual)

Core Entities:

1. User (Student, Alumni, Admin)
2. Company
3. Question
4. Response
5. WorkHistory
6. TrustScore
7. Referral
8. CompanyInsight
9. Notification

Relationships:

- Student [1] ——— [Many] Question
- Question [1] ——— [1] Response
- Alumni [1] ——— [Many] Response
- Alumni [1] ——— [Many] WorkHistory
- Company [1] ——— [Many] WorkHistory

- Question [Many] ——— [1] Company
- Student [1] ——— [Many] Referral
- Alumni [1] ——— [Many] Referral (as reviewer)
- Alumni [1] ——— [1] TrustScore
- Company [1] ——— [1] CompanyInsight

10.2 Key Database Tables (Schema Overview)

Table: users Purpose: Store all user types with role differentiation

- id (PK, INT, AUTO_INCREMENT)
- email (VARCHAR(120), UNIQUE, NOT NULL)
- password_hash (VARCHAR(255), NOT NULL)
- name (VARCHAR(100), NOT NULL)
- role (ENUM: 'student', 'alumni', 'admin')
- is_verified (BOOLEAN, DEFAULT FALSE)
- created_at (TIMESTAMP)

Table: students Purpose: Student-specific profile information

- user_id (PK, FK → users.id)
- roll_number (VARCHAR(20), UNIQUE)
- batch_year (INT)
- branch (VARCHAR(50))
- current_year (INT: 1-4)
- cgpa (DECIMAL(3,2), NULL)
- show_cgpa (BOOLEAN, DEFAULT TRUE)
- profile_completeness (INT, 0-100)
- career_goals (TEXT)

Table: alumni Purpose: Alumni mentor profiles

- user_id (PK, FK → users.id)
- batch_year (INT)
- branch (VARCHAR(50))

- current_company_id (FK → companies.id)
- current_role (VARCHAR(100))
- linkedin_url (VARCHAR(200))
- is_accepting_questions (BOOLEAN, DEFAULT TRUE)
- max_questions_per_month (INT, DEFAULT 10)
- trust_score (DECIMAL(5,2), DEFAULT 50.0)
- total_questions_answered (INT, DEFAULT 0)
- avg_response_time_hours (DECIMAL(5,2))
- verification_status (ENUM: 'pending', 'approved', 'rejected')

Table: companies Purpose: Company master data

- id (PK, INT, AUTO_INCREMENT)
- name (VARCHAR(100), UNIQUE, NOT NULL)
- website (VARCHAR(255))
- logo_url (VARCHAR(255))
- industry (VARCHAR(50))
- company_type (ENUM: 'Startup', 'Mid-size', 'Large', 'MNC')
- headquarters (VARCHAR(100))
- total_alumni_count (INT, DEFAULT 0) # Denormalized
- active_mentor_count (INT, DEFAULT 0) # Denormalized
- avg_trust_score (DECIMAL(5,2)) # Denormalized

Table: work_history Purpose: Track alumni employment history

- id (PK, INT, AUTO_INCREMENT)
- alumni_id (FK → alumni.user_id, NOT NULL)
- company_id (FK → companies.id, NOT NULL)
- role (VARCHAR(100), NOT NULL)
- location (VARCHAR(100))
- start_date (DATE, NOT NULL)
- end_date (DATE, NULL) # NULL if current

- is_current (BOOLEAN, DEFAULT FALSE)
- tech_stack (JSON) # ["Python", "PostgreSQL"]

Table: questions Purpose: Student questions

- id (PK, INT, AUTO_INCREMENT)
- student_id (FK → students.user_id, NOT NULL)
- company_id (FK → companies.id, NOT NULL)
- assigned_mentor_id (FK → alumni.user_id, NULL)
- question_type (ENUM: 'interview_prep', 'roadmap', 'culture', 'salary', 'resume', 'referral', 'general')
- question_text (TEXT, NOT NULL)
- context_json (JSON) # Student background, current status
- urgency (ENUM: 'normal', 'high', 'urgent', DEFAULT 'normal')
- status (ENUM: 'pending', 'queued', 'in_progress', 'answered', 'closed')
- queue_position (INT, NULL)
- submitted_at (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP)
- answered_at (TIMESTAMP, NULL)

Table: responses Purpose: Mentor answers

- id (PK, INT, AUTO_INCREMENT)
- question_id (FK → questions.id, UNIQUE, NOT NULL)
- mentor_id (FK → alumni.user_id, NOT NULL)
- response_text (TEXT, NOT NULL)
- resources_json (JSON) # Links, attachments
- responded_at (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP)
- last_edited_at (TIMESTAMP, NULL)

Table: feedback Purpose: Outcome-based feedback from students

- id (PK, INT, AUTO_INCREMENT)
- question_id (FK → questions.id, UNIQUE, NOT NULL)
- was_relevant (BOOLEAN)

- was_implemented (ENUM: 'yes', 'partially', 'no', 'too_early')
- outcome (ENUM: 'got_job', 'interview_calls', 'skills_improved', 'somewhat_helpful', 'not_helpful')
- rating (INT, 1-5)
- comments (TEXT, NULL)
- submitted_at (TIMESTAMP)

Table: trust_scores Purpose: Calculated trust scores per mentor

- mentor_id (PK, FK → alumni.user_id)
- current_score (DECIMAL(5,2), DEFAULT 50.0)
- total_responses (INT, DEFAULT 0)
- positive_outcomes (INT, DEFAULT 0)
- negative_outcomes (INT, DEFAULT 0)
- response_rate (DECIMAL(3,2)) # 0.00-1.00
- last_calculated_at (TIMESTAMP)

Table: referrals Purpose: Student referral requests

- id (PK, INT, AUTO_INCREMENT)
- student_id (FK → students.user_id, NOT NULL)
- mentor_id (FK → alumni.user_id, NOT NULL)
- company_id (FK → companies.id, NOT NULL)
- target_role (VARCHAR(100))
- portfolio_links_json (JSON)
- resume_url (VARCHAR(255))
- student_message (TEXT)
- status (ENUM: 'pending', 'approved', 'declined', 'changes_requested')
- mentor_feedback (TEXT, NULL)
- created_at (TIMESTAMP)
- reviewed_at (TIMESTAMP, NULL)

Table: company_insights Purpose: Aggregated company intelligence

- company_id (PK, FK → companies.id)
- typical_hiring_months (JSON) # ["January", "July"]
- avg_hiring_duration_days (INT)
- fresher_salary_min (INT, NULL)
- fresher_salary_max (INT, NULL)
- interview_difficulty (INT, 1-10)
- interview_rounds_json (JSON) # Array of round objects
- must_have_skills (JSON)
- tech_stack_json (JSON)
- work_life_balance_rating (DECIMAL(3,2))
- remote_policy (ENUM: 'remote', 'hybrid', 'office')
- last_updated (TIMESTAMP)

Table: notifications Purpose: In-app notifications

- id (PK, INT, AUTO_INCREMENT)
- user_id (FK → users.id, NOT NULL)
- type (ENUM: 'info', 'action_required', 'success', 'warning')
- title (VARCHAR(200))
- message (TEXT)
- link_url (VARCHAR(255), NULL)
- is_read (BOOLEAN, DEFAULT FALSE)
- created_at (TIMESTAMP)

Table: student_skills Purpose: Student skill tags (many-to-many)

- id (PK, INT, AUTO_INCREMENT)
- student_id (FK → students.user_id)
- skill_name (VARCHAR(50))
- proficiency (ENUM: 'beginner', 'intermediate', 'advanced')

Table: alumni_expertise Purpose: Alumni expertise tags (many-to-many)

- id (PK, INT, AUTO_INCREMENT)

- alumni_id (FK → alumni.user_id)
- skill_name (VARCHAR(50))

Table: student_target_companies Purpose: Student's wishlist companies

- id (PK, INT, AUTO_INCREMENT)
- student_id (FK → students.user_id)
- company_id (FK → companies.id)
- priority (INT, 1-5) # 1 = highest
- added_at (TIMESTAMP)

10.3 Indexing Strategy

Purpose: Optimize query performance for frequent operations

Indexes to Create:

- users(email) - UNIQUE # Fast login lookup
- students(roll_number) - UNIQUE
- alumni(current_company_id) # Company-alumni filtering
- work_history(alumni_id, company_id) # Mentor matching
- work_history(company_id) # Company page queries
- questions(student_id, status) # Student dashboard
- questions(assigned_mentor_id, status) # Mentor queue
- questions(company_id, status) # Company Q&A archive
- responses(question_id) # O(1) response lookup
- notifications(user_id, is_read, created_at) # Notification feed

10.4 Data Integrity Constraints

Foreign Key Constraints:

- ON DELETE CASCADE: notifications (when user deleted)
- ON DELETE SET NULL: questions.assigned_mentor_id (if mentor account deleted)
- ON DELETE RESTRICT: work_history.company_id (prevent company deletion if history exists)

Check Constraints:

- students.cgpa BETWEEN 0.00 AND 10.00
- alumni.trust_score BETWEEN 0.00 AND 100.00
- questions.urgency IN ('normal', 'high', 'urgent')
- feedback.rating BETWEEN 1 AND 5

Unique Constraints:

- (student_id, company_id) in student_target_companies # No duplicate wishlist entries
- question_id in responses # One response per question
- question_id in feedback # One feedback per question

---

11. User Interface Specifications

11.1 Design Principles

Visual Design:

- **Color Scheme:**
    - Primary: #2563EB (Blue) - Trust, professionalism
    - Secondary: #10B981 (Green) - Success, positive outcomes
    - Accent: #F59E0B (Amber) - Highlights, calls-to-action
    - Neutral: #6B7280 (Gray) - Text, backgrounds
    - Error: #EF4444 (Red) - Alerts, warnings
- **Typography:**
    - Headings: Inter or Poppins (Sans-serif, modern)
    - Body: System fonts (-apple-system, BlinkMacSystemFont, "Segoe UI")
    - Code: 'Courier New', monospace
- **Spacing:** 8px grid system (8px, 16px, 24px, 32px, 48px)
- **Border Radius:** 8px for cards, 4px for buttons

UX Principles:

- **Clarity:** Clear labels, no jargon
- **Consistency:** Same UI patterns across pages

- **Feedback:** Visual confirmation for all actions
- **Efficiency:** Minimize clicks to complete tasks
- **Accessibility:** Keyboard navigation, screen reader support

11.2 Page Layouts

11.2.1 Navigation Bar (Global) Present on all pages

Desktop Layout:

```
┌──────────────────────────────────────────────────────┐
│ [ASCEND Logo]  Companies  Mentors  Knowledge Base │
│                                    │
│                  [🔍 Search] [🔔 3] [Profile▼]│
└──────────────────────────────────────────────────────┘
```

Mobile Layout (<768px):

```
┌──────────────────────────────────────────────────────┐
│ [☰]  ASCEND              [🔔 3] [Profile▼]   │
└──────────────────────────────────────────────────────┘
```

(Collapsible menu on hamburger click)

Components:

- **Logo:** Clickable, returns to dashboard
- **Main Links:** Companies, Mentors (alumni directory), Knowledge Base
- **Search Bar:** Global search (desktop only, icon on mobile)
- **Notifications:** Bell icon with badge count
- **Profile Dropdown:**
    - My Profile
    - My Questions (students) / My Queue (mentors)
    - Settings
    - Logout

**11.2.2 Student