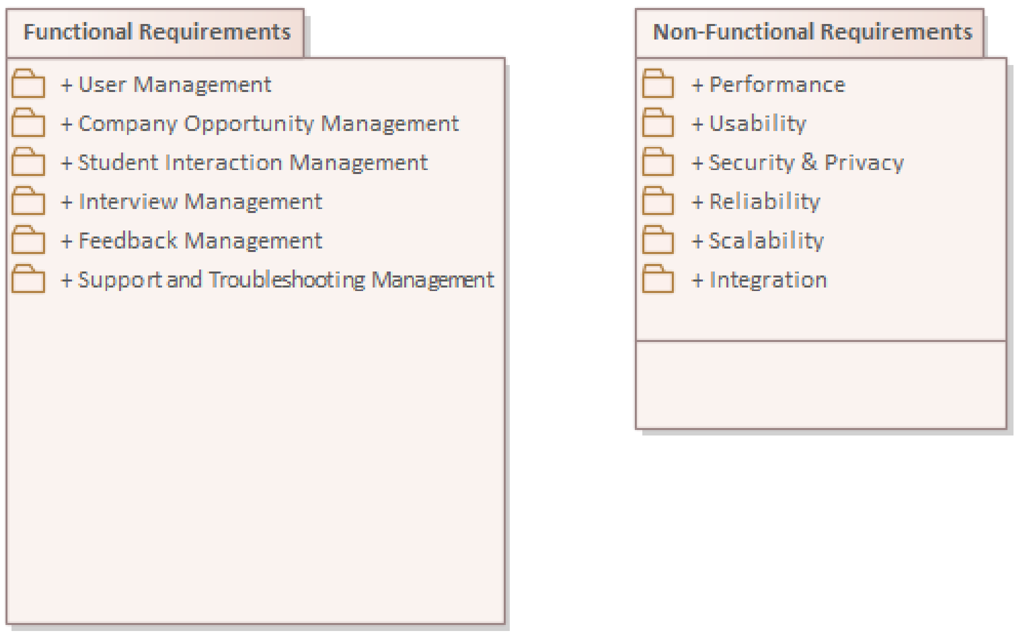
3.3 User Requirements

This section outlines the core functional and experiential needs of the target users, primarily students and recent graduates, based on identified limitations in existing platforms and observed user behavior patterns. These requirements will guide the design and development of our AI-powered opportunity matching system.



Identified User Requirements:

- **Personalized Opportunity Matching:**
  Users require intelligent job and internship suggestions that align with their education, interests, skills, and location preferences, without the need to manually filter through generic listings.

- **Conversational Interface & Real-Time Interaction:**
  Students prefer interactive and responsive environments. A conversational AI embedded in a familiar platform (such as Discord) offers a more engaging and intuitive experience compared to static web forms.

- **Internship and Entry-Level Focus:**
  Users seek platforms that prioritize early-career roles, especially internships, part-time positions, and graduate opportunities—roles often underserved or hard to locate on mainstream job boards.

- **Community and Peer Engagement:**
  Access to a peer network or community for sharing opportunities, asking questions, and receiving support is a valuable feature, especially for first-time job seekers navigating the career space.

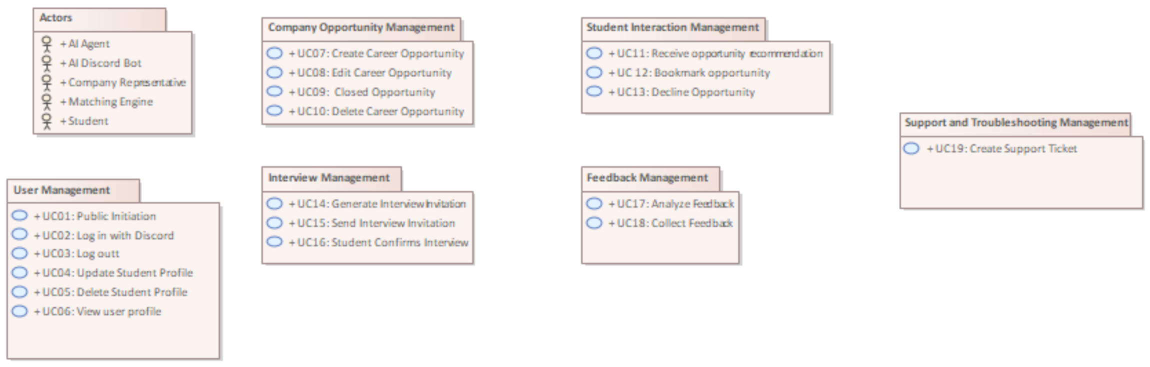
- **User-Controlled Application Workflow:**
  Unlike fully automated systems, users desire visibility and control over where and how they apply. They expect the platform to suggest high-quality roles but allow them to make the final decision.

- **Live Notifications and Alerts:**
  Users want to be notified immediately when new opportunities that match their profile become available. Timely alerts reduce the chance of missing deadlines and improve responsiveness.

- **Simple, Accessible Interface:**
  The platform should be mobile-friendly, quick to set up, and seamlessly integrate with platforms students already use (such as Discord or email), requiring no technical expertise.

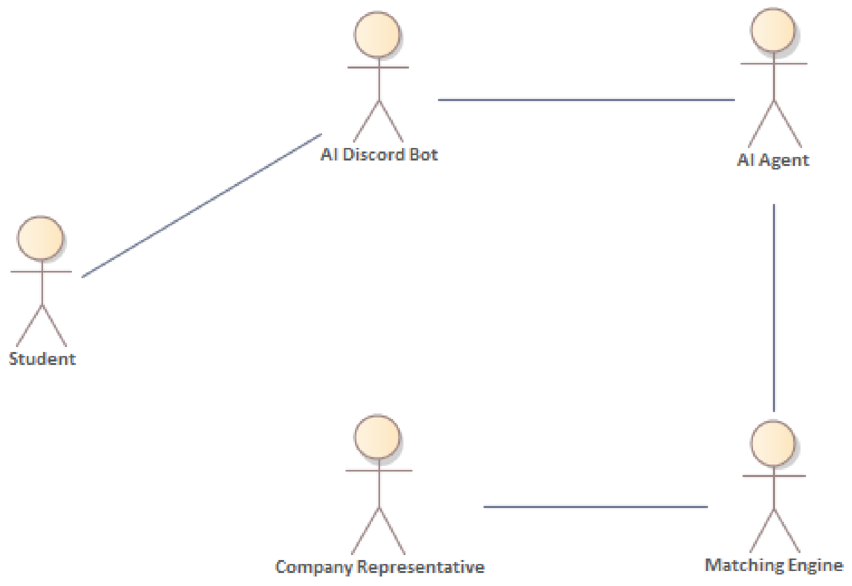
3.1 Functional Requirements

In this section, the functional requirements of the AI-Powered Career Opportunity Distribution System are defined in the form of use cases, reflecting the exact interactions between actors and the system.



Actors
+AI Agent
+AI Discord Bot
+Company Representative
+Matching Engine
+Student
Company Opportunity Management
+UC07: Create Career Opportunity
+UC08: Edit Career Opportunity
+UC09: Closed Opportunity
+UC10: Delete Career Opportunity
Student Interaction Management
+UC11: Receive opportunity recommendation
+UC 12: Bookmark opportunity
+UC13: Decline Opportunity
Support and Troubleshooting Management
+UC19: Create Support Ticket
User Management
+UC01: Public Initiation
+UC02: Log in with Discord
+UC03: Log outt
+UC04: Update Student Profile
+UC05: Delete Student Profile
+UC06: View user profile
Interview Management
+UC14: Generate Interview Invitation
+UC15: Send Interview Invitation
+UC16: Student Confirms Interview
Feedback Management
+UC17: Analyze Feedback
+UC18: Collect Feedback

3.1.1 Actors



Student:
A user registered in Discord who manages their profile, receives career opportunities, and interacts with opportunities.

AI Discord Bot:
The intelligent, automated agent that manages interactions and matches students to job opportunities, integrated with Discord.

AI Agent:
Service that reformats conversation data and orchestrates the Matching Engine.

Matching Engine:

Component that evaluates a student profile against the full opportunity index and returns a ranked list of matches.

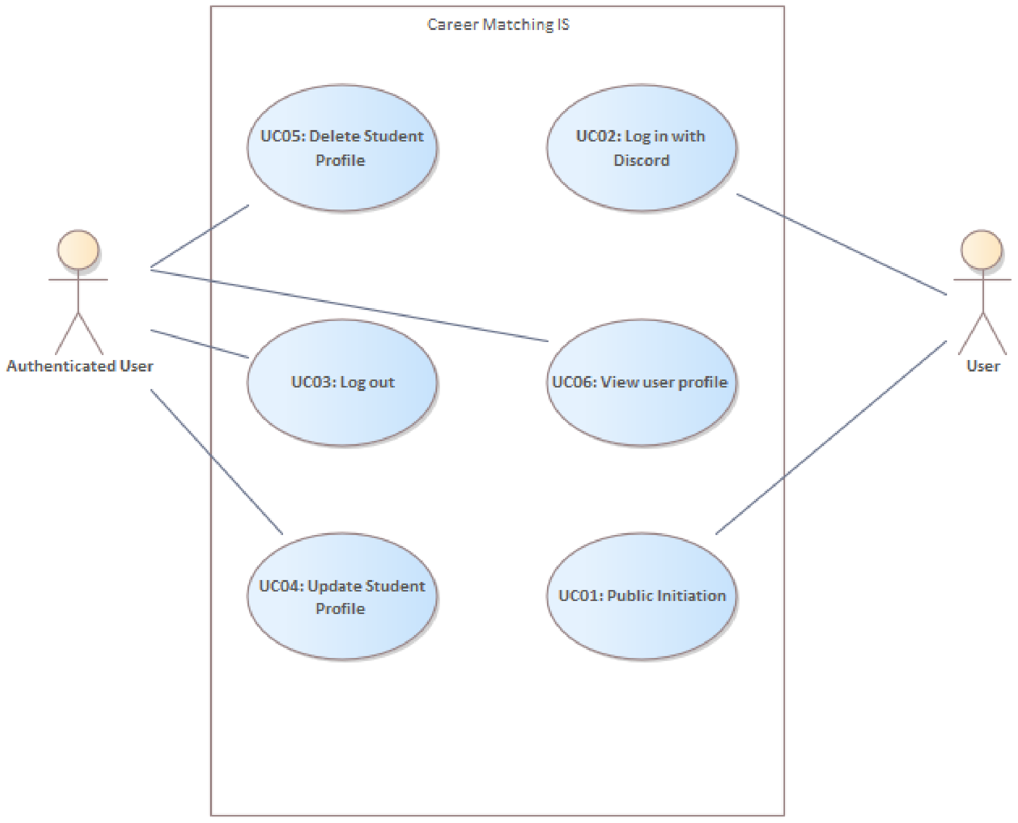
Company Representative:

Posts, edits or withdraws opportunities on EXPERTS.AI; changes are picked-up asynchronously by the Matching Engine.

3.1.2 User Management

This package defines the use cases related to basic user management.

The user management use cases cover signing in to the system through Discord (or signing out), creating an account, viewing personal details, and editing or deleting the user profile.



3.1.2.1 UC01: Public Initiation

The user triggers interaction with the bot in any public server channel by typing “/start” (or a recognised greeting such as “Hi bot”). The command must be visible to the bot and to other users in the channel, yet no subsequent personal data should appear in public. The bot opens a private DM thread with the student and sends a welcome banner plus a short explanation of the service.

3.1.2.2 UC02: Log in with Discord

After collecting the necessary data (name, skills, career interest, resume) a short welcome message confirms sign-in.

Basic path: logging in

1. Bot asks for the student’s name; validates that it is not empty.
2. Prompts for skills → student selects tags from a predefined list or types custom ones.
3. Prompts for preferred opportunity type; validates choice.
4. Requests a short free-text statement on interests; length limited to 500 chars.
5. Offers file-upload; accepts PDF, DOCX, or a URL.
6. Displays a summary of the captured profile and asks for confirmation (Save / Edit / Cancel).
7. On Save, the profile is persisted; the student receives a “Profile complete” confirmation and the bot queues the profile for matching.

Alternative path: Missing data

At any step the student might forget to or choose not to add a data, the bot records “unknown” and continues, or asks the user to fill the data.

Alternative path: Cancelling

Student types /cancel; bot abandons onboarding and deletes un-saved data.

3.1.2.3 UC03: Log out

The authenticated user can end the session at any time by sending the /logout command (or by removing the bot’s OAuth permission inside Discord’s *Authorized Apps* panel)

3.1.2.4 UC04: Update Student Profile

Users can refine their profile at any time. The bot opens an interactive DM wizard where they can adjust:

- Their skills,
- desired opportunity types (internship, thesis, part-time, full-time),

- preferred locations or remote-only flag,
- CV or portfolio URL,
- notification preferences (real-time, daily digest, mute).

3.1.2.5 UC05: Delete Student Profile

A user who no longer wishes to use the service can delete their profiles. The bot presents a two-step confirmation to prevent accidental loss of data.

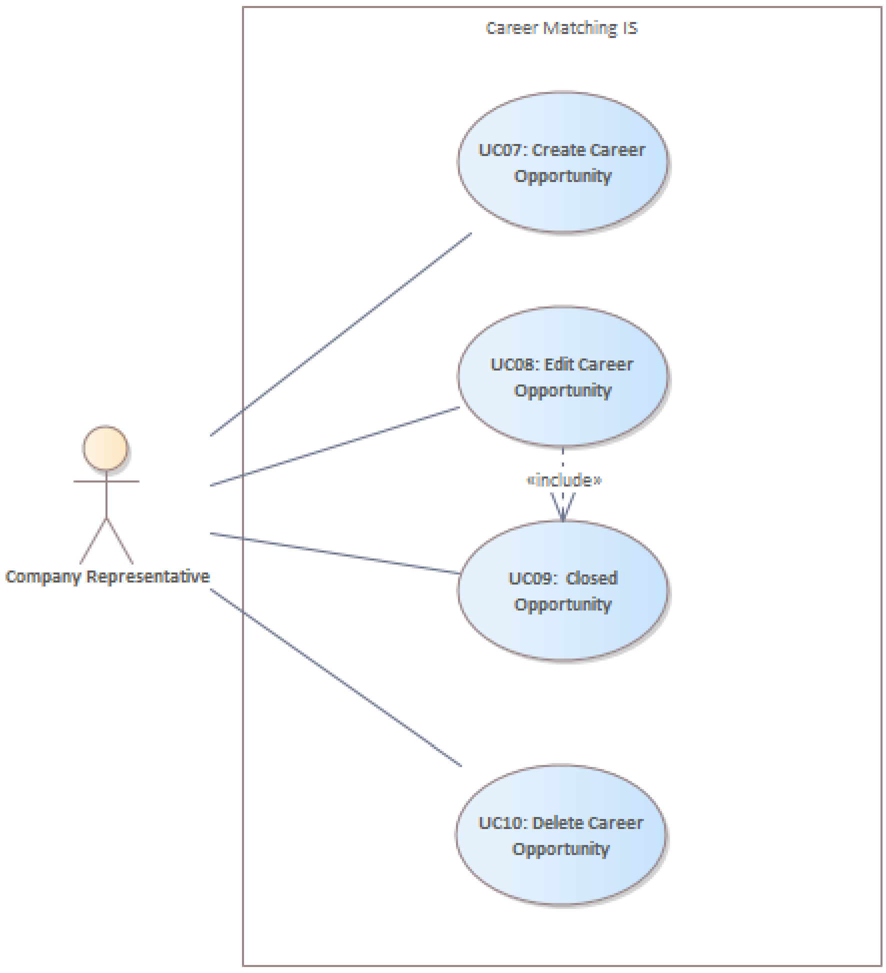
3.1.2.6 UC06: View user profile:

By issuing /profile view, the user receives a compact, embed-style snapshot that lists:

• Registered e-mail
• Skill summary
• Current opportunity preferences
• Number of bookmarked roles and interviews in progress

3.1.3 Company Opportunity Management

This package groups all use-cases that enable a company representative (acting through the EXPERTS.AI and Discord bot) to create, publish, maintain and ultimately close an opportunity. The package also covers interactions with matched students, viewing bookmarked candidates, issuing interview invitations and recording hiring outcomes.


Career Matching IS
UC07: Create Career Opportunity
UC08: Edit Career Opportunity
«include»
UC09: Closed Opportunity
UC10: Delete Career Opportunity
Company Representative

3.1.3.1 UC07: Create Career Opportunity

Company representatives add new opportunities detailing required skills, roles, and deadlines to EXPERTS.AI

3.1.3.2 UC08: Edit Career Opportunity

Company representatives update existing opportunities' descriptions, requirements, or application deadlines.

3.1.3.3 UC09: Closed Opportunity

When the position is filled or withdrawn, the bot flips the status to *closed*, stops new recommendations, and sends a courtesy DM to every student who bookmarked the opening, thanking them and indicating the outcome (“filled”, “cancelled”, etc.).

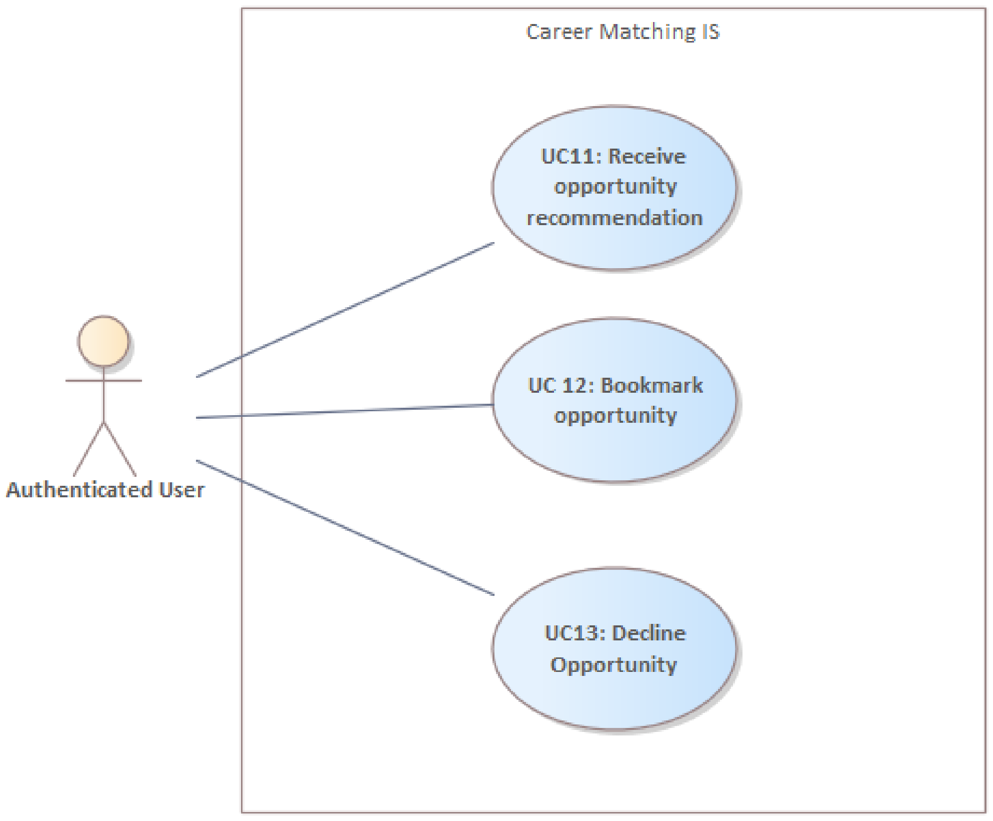
3.1.3.4 UC10: Delete Career Opportunity

Representatives remove opportunities that are no longer active or needed.The action requires confirmation; once executed, the record is purged and no history is kept except an audit log stub for compliance.

If critical fields change by the company representative (title, location, tech stack), the system re-runs the matcher and, when appropriate, notifies new students while retracting the card from students who no longer match.

3.1.4 Student Interaction Management

This package defines the use cases that let a user receive, review and manage career opportunities delivered by the Discord bot.


Career Matching IS
UC11: Receive opportunity recommendation
UC 12: Bookmark opportunity
UC13: Decline Opportunity
Authenticated User

3.1.4.1 UC11: Receive opportunity recommendation

The system sends a personalised opportunity card to the student via Discord DM as soon as a student's skills and career interest meets a job offer.

Basic Path: Matching Opportunity with Student

1. Discord Bot acknowledges the request and loads the student's stored profile
2. Bot sends the profile to the AI Agent.
3. AI Agent forwards profile vector to the Matching Engine

4. Matching Engine evaluates the request and matches with the job offers.
5. Job offer has been found.
6. Bot composes a rich message containing title, company, location, duration, tech stack and brief description, plus buttons: Bookmark, see more, decline.

7. System delivers the message to the student's DM.

8. Student reads the message and chooses one of the available actions. (Bookmark, See more, Decline.)

Alternative Path: More opportunities

1. Student reacts "see more."

2. Bot requests the next item in the ranked list (repeat basic steps 6 – 9).

3. Control returns to Basic Path step 8 for the new card.

Exception: No suitable opportunity

1. Matching Engine returns an empty list.

2. AI Agent notifies Bot; Bot apologies, offers to broaden criteria, and closes the card.

3.1.4.2 UC 12: Bookmark opportunity

After an opportunity card is presented, the student can bookmark it to express interest.The system must persist the decision, notify the company, and prepare the ground for a later interview.

Basic Path: Bookmark Opportunity

1. Student bookmarks an opportunity

2. Discord Bot confirms and removes any reaction-based menus from the card.

3. Bot forwards the student's profile and the bookmarked opportunity ID to the AI Agent for persistence and analytics.

4. AI Agent sets the student-opportunity relation to "interested", and creates an Interview Draft (status = pending) that will be processed later by the *Interview*

Management package.

5. Bot informs the student that their profile was sent to the company.

Exception: Already Bookmarked

Occurs at step 1 if a duplicate bookmark is detected.

1. Bot responds “You’ve already bookmarked this role.”

2. No further actions; return to waiting for other reactions. If no response is met, bot cancels the bookmarked job offer.

3.1.4.3 UC13: Decline Opportunity

The student explicitly rejects a delivered opportunity card. The system must acknowledge the choice, log it for learning metrics and offer next-step options, while ensuring the same card is not re-sent unless the student later revokes the decision.

Basic Path: Rejecting the Opportunity

1. Student declines the opportunity.

2. Discord Bot records the rejection, then sends a reject request to the AI Agent.

3. AI Agent forwards the request to the Matching Engine along with the current profile vector.

4. Matching Engine marks the opportunity as declined for this student and lowers its ranking weight in future matching.

5. Bot sends a confirmation DM for rejection.
6. Bot asks whether the student wants to see another match, or give feedback to redo the matching based on the feedback.

7. Student either agree to see more opportunities, skip or undo.

8. Flow ends:

    - If Yes, show next → control returns to *UC11 Receive Opportunity Recommendation* (step 6) for the next opportunity card.

    - If No → DM thread closes; bot waits for the next matching trigger.

Alternative Path A: Change of Mind (Undo Decline)

1. At step 7 the student types “/undo” (or presses a contextual “Undo” button.)
2. Bot cancels the decline flag in the Matching Engine, restores the card.

Alternative Path B: Provide Detailed Feedback

1. Student wants to give feedback for the current opportunity card.
2. Bot opens a short free-text prompt..
3. Student submits a comment (e.g., “Salary range not shown”).
4. Bot stores the comment, thanks the student, then continues to do the matching again considering the given feedback.

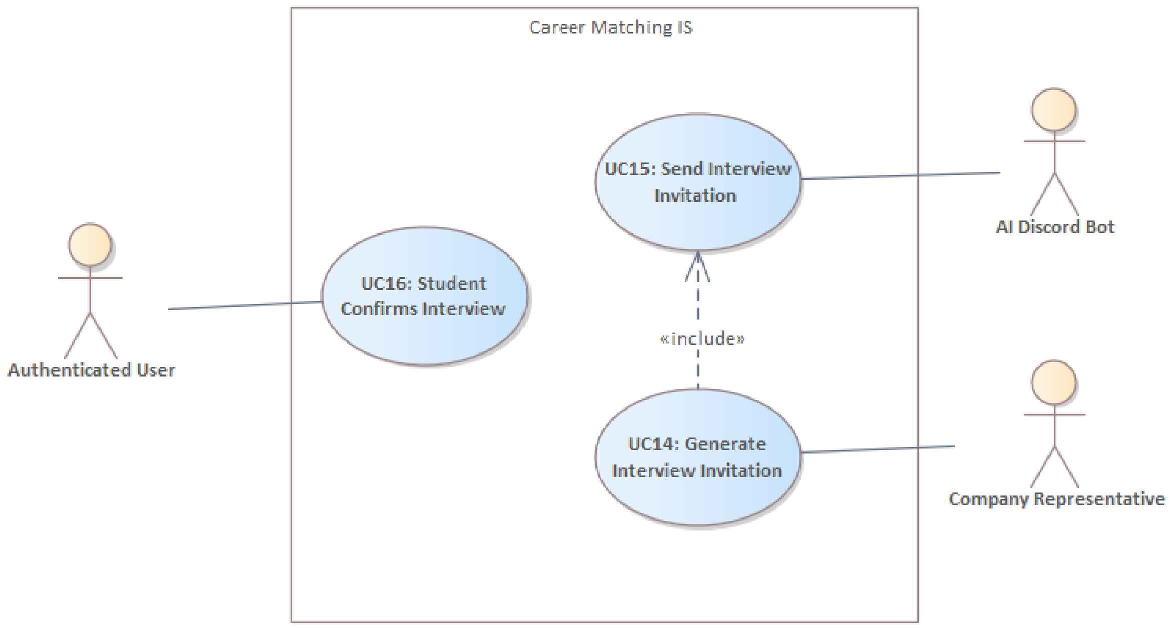
Exception Path: Feedback Timeout

If the student does not respond within 60 seconds

1. Bot sends a reminder.
2. After additional 60 seconds with no interaction, the bot records no-feedback and proceeds to give other job opportunities.

3.1.5 Interview Management

This package covers the flow that begins when a company decides to interview a book-marked student and ends when the interview slot is either confirmed, rescheduled, or cancelled.


Career Matching IS
UC15: Send Interview Invitation
AI Discord Bot
UC16: Student Confirms Interview
Authenticated User
«include»
UC14: Generate Interview Invitation
Company Representative

3.1.5.1 UC14: Generate Interview Invitation

Company Representative enters the date, time, format, and interviewer; system creates a draft invitation.

3.1.5.2 UC15: Send Interview Invitation

Discord Bot delivers the invitation to the student, stores a pending status, and starts a response timer.

3.1.5.3 UC16: Student Confirms Interview

Student accepts; bot schedules reminders and updates status to confirmed.

Basic Path: Confirmation of Interview

1. Student clicks Confirm in the Discord interview card.
2. Discord Bot sets interview status to confirmed.
3. AI Agent records the event in the student timeline for analytics.
4. System emits a real-time webhook to the Company Representative: “Candidate confirmed.”
5. Company representative arranges an optimal date and time for the interview.
6. Matching Engine stores and sends to AI Engine.
7. Discord Bot acknowledges the optimal dates and times and sends a message to the user.
8. User selects a time for the interview and types “confirm”.
9. Bot confirms the interview and sends an optional “Interview Prep” message (tips, company link, sample questions).

Exception: Double-booking
If a slot has been taken by another candidate before confirmation.

1. Bot detects clash, reverts status to pending, apologies.

2. Bot starts to check for other optimal dates.

Exception: No suitable interview

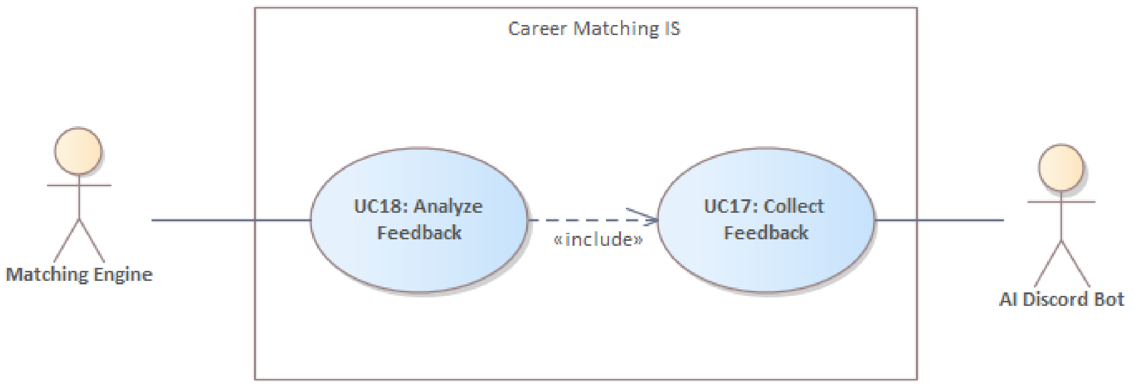
1. Matching Engine returns an empty list.
2. AI Agent notifies Bot; Bot apologies, offers for other job opportunities.

Exception: No response in a 3-day response time

1. Company representative cancels the interview.
2. Matching Engine closes that job offer for the user.
3. AI Agent notifies the bot and sends a message to the user telling that the desired interview can no longer be taken.

3.1.6 Feedback Management

Allow the student to provide feedback about recommended opportunities and interview experiences, enabling the matching algorithm to continuously learn and improve recommendations.



3.1.6.1 UC17: Collect Feedback

Post-interview or after reviewing an opportunity, the student provides structured feedback about the relevance of the opportunity or the quality of the interview process. This helps the system better understand how accurate the match was and how helpful the opportunity turned out to be.

Basic Path: Collecting Feedback

1. After a student bookmarks or completes an interview, the Discord Bot automatically sends a message asking:

    - “How relevant was this opportunity for you?” (if bookmarked or declined)

2. The Student responds by selecting a rating on a 5-point scale:
   1 = Not relevant / Poor
   2 = Somewhat relevant / Below average
   3 = Neutral
   4 = Relevant / Good
   5 = Perfect match / Excellent

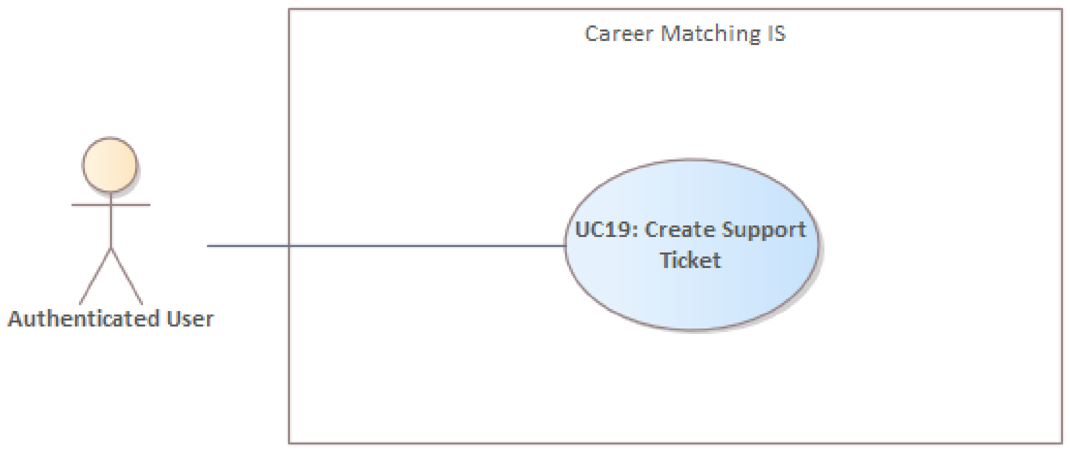
3. The Bot confirms stores the rating in the feedback database and updates profile preferences.

3.1.6.2 UC18: Analyze Feedback

All collected feedback data is automatically analyzed and used to improve the relevance and accuracy of future recommendations. The Matching Engine updates its scoring weights and matching threshold based on trends and patterns from recent feedback. Next time the student requests a match, the Discord Bot uses the refined scoring from the updated Matching Engine.

3.1.7 Support and Troubleshooting Management

This package defines the use cases related to reporting, managing, and resolving support issues encountered by students while interacting with the Discord Bot or any connected services.



It includes ticket creation, automated first-line troubleshooting, and escalation paths for unresolved or complex problems.

3.1.7.1 UC19: Create Support Ticket

Students report technical issues or unexpected behavior directly through the Discord Bot, which generates a structured support ticket for follow-up.

Basic Path:

1. Student types the command /support or selects the “Need Help” button in the bot interface.

2. Discord Bot opens an interactive form asking the student to describe the problem, optionally attach a screenshot or paste an error code.

3. Upon submission, the bot confirms and stores a new ticket record.
4. If the issue matches a known pattern or FAQ, the bot sends an automatic response with a possible solution.

5. If no automatic solution is found, ticket data is logged in the system and added to the troubleshooting queue.
6. A system administrator reviews the ticket, investigates the issue, and provides a solution or further instructions.
7. Once the resolution is submitted, the bot notifies the student

Alternative Path: Anonymous Request

If the student is not authenticated when using “/support”.

1. Bot responds with “You must sign in first to report an issue.”

2. Redirects the student to the login flow and resumes ticket creation after successful sign-in.

Exception Path: Invalid Input

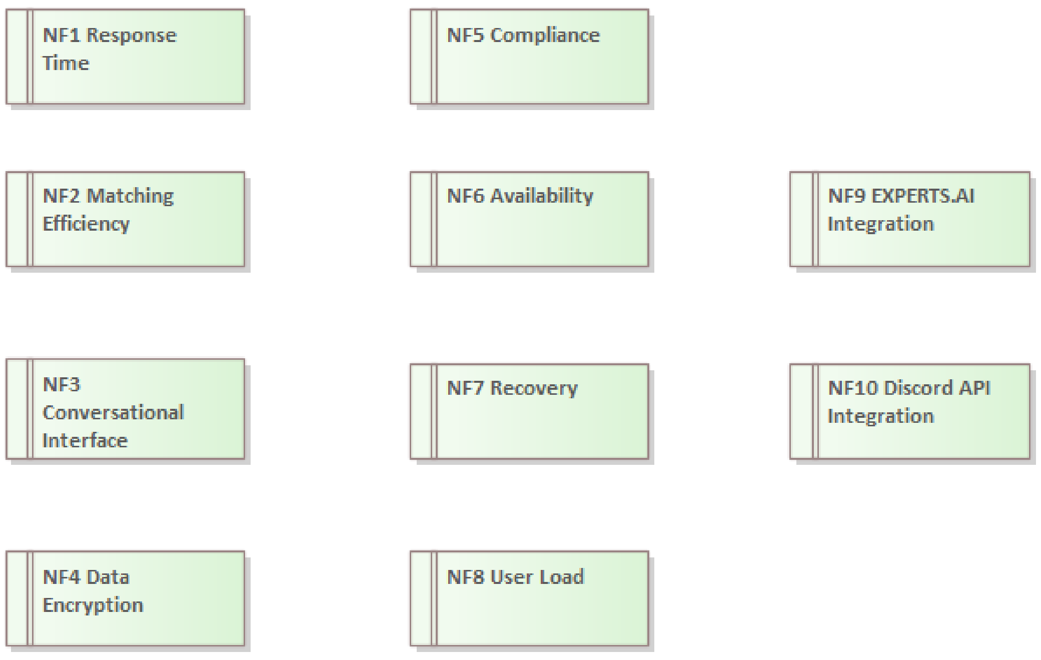
Student submits an incomplete form (e.g., blank description):

1. Bot displays inline validation error.

2. Student is prompted to revise and resubmit.

3.2 Non-Functional Requirements

This section specifies measurable qualities and constraints on the system’s behavior, which ensure usability, reliability, and performance of the AI-powered Career Opportunity Distribution System.



3.2.1 NF1 Response Time:

The system shall respond to user interactions within 3-5 seconds at most.

3.2.2 NF2 Matching Efficiency:

Opportunity-to-student matching must be completed within 10 seconds from the moment a new opportunity is posted.

3.2.3 NF3 Conversational Interface:

The system interactions shall be structured clearly and understandably, such that any primary interaction (e.g., profile creation, expressing interest, viewing details) requires no more than 3 interaction steps from start to finish. The clarity will be evaluated based on usability testing and feedback questionnaires:

1. At least 80% of test users must successfully complete given interaction tasks without additional help.
2. User satisfaction surveys must indicate a minimum average rating of 4 out of 5 regarding clarity and ease of use.

3.2.4 NF4 Data Encryption:

All user CVs and personal data shall be encrypted using AES-256 encryption standards at rest and during transmission.

3.2.5 NF5 Compliance:

The system must adhere strictly to GDPR standards for data privacy, ensuring explicit user consent, transparency in data use, and secure data management practices. Compliance shall be verified by regular internal audits conducted quarterly.

3.2.6 NF6 Availability:

The system shall ensure 99% availability. To achieve this, the proposed architecture includes:

1. Redundant cloud-based infrastructure (AWS, Azure, or similar).
2. Automatic scaling to handle variable loads and ensure stability during peak usage.
3. Failover mechanisms that redirect traffic automatically in case of partial system failure.
4. Regular backups to enable rapid recovery.

3.2.7 NF7 Recovery:

In case of unexpected downtime, the system shall automatically detect issues and restore functionality within 30 minutes using automated scripts and predefined recovery protocols.

3.2.8 NF8 User Load:

The system must comfortably support concurrent users actively interacting with the Discord bot and the web application without noticeable degradation in performance

3.2.9 NF9 EXPERTS.AI Integration:

The integration between the system and EXPERTS.AI must guarantee reliable data synchronization. This will be validated through regular synchronization tests conducted monthly, ensuring no loss of information or delay in data transfer.

3.2.10 NF10 Discord API Integration:

The integration with Discord's API must be stable. The reliability will be monitored continuously through system logs and monthly reporting.