Chapter 3

proposed system's

overview

The ASTU Almunilink system is designed to create a comprehensive platform that connects alumni, current students, faculty, and companies associated with the Adama Science and Technology University (ASTU). The primary goal of this system is to enhance networking opportunities, facilitate job placements, and foster mentorship relationships within the university community. By leveraging modern web technologies, the system aims to provide a user-friendly, secure, and efficient environment for all stakeholders.

Objectives

- **Networking:** To enable alumni and students to connect with each other, fostering relationships that can lead to mentorship and professional growth.
- **Job Opportunities:** To provide a centralized platform for job postings and internships that are accessible to students and alumni, enhancing their career prospects.
- **Mentorship:** To create avenues for experienced alumni to offer guidance and support to current students, helping them navigate their academic and professional journeys.
- **Collaboration:** To facilitate collaboration between faculty and alumni on research projects and academic initiatives, enriching the educational experience at ASTU.
- **Company Engagement:** To connect companies with potential candidates from the university, streamlining recruitment processes and allowing for tailored job postings.

Key Features

- **User Roles and Management: -** The system supports multiple user roles: Admin, Alumni, Students, Faculty, and Companies. Each role has specific permissions and functionalities tailored to their needs.
- **User Registration and Profile Management: -** Users can register and create profiles, detailing their educational background, work experience, and skills. Profiles can be edited and updated as needed.

- **Job and Opportunity Listings: -** Alumni and companies can post job openings and internships, which students can browse and apply for directly through the platform.
- **Networking Tools: -** The platform allows users to send connection requests, communicate through a messaging system, and participate in networking events.
- **Event Management: -** Users can create, promote, and RSVP to events such as workshops, seminars, and networking sessions. Admins review and approve events before they are published.
- **Feedback and Rating: -** Users can provide feedback on events, job applications, and mentorship experiences, contributing to a community rating system.
- **Admin Controls: -** Admins have comprehensive control over user management, including the ability to verify registrations, monitor user activity, and generate reports on system usage.

Technical Architecture

The ASTU Almunilink system will be built using a modern web stack that includes:

- **Frontend**: A responsive web application developed using frameworks such as React or Angular to ensure a seamless user experience across devices.
- **Backend**: A robust server-side application using Node.js, Django, or Ruby on Rails to handle business logic, user authentication, and data management.
- **Database**: A relational database such as PostgreSQL or MySQL to store user data, job listings, events, and application details.

Non-Functional Considerations

The system will adhere to non-functional requirements that ensure quality, performance, and security:

- **Performance**: The application will be optimized to handle multiple concurrent users with minimal response times.
- **Security**: User data will be protected through encryption, secure authentication methods (including 2FA), and regular security audits.
- **Usability**: The user interface will be designed to be intuitive and accessible, accommodating users of varying technical expertise.

- **Scalability**: The architecture will allow for horizontal scaling to accommodate growth in user numbers and data volume

3.4 System Model

This section presents the system model for the **ASTU Almunilink** system, illustrating its structure, functions, behavior, and interactions among various components. The model encompasses conceptual, logical, and physical aspects, providing a comprehensive view of how the system operates.

1. Conceptual Model

The conceptual model provides a high-level view of the system, focusing on the key entities, relationships, and functionalities without delving into technical specifics. It serves as a foundation for understanding the system's purpose and structure.

Key Entities: -

User: Represents individuals using the platform, including alumni, students, faculty, companies, and admins.

Profile: Contains personal and professional information for each user. -

Job: Represents job postings made by companies or alumni, including details such as title, description, and requirements. -

Event: Represents networking or educational events created by users, including date, time, and location. -

Connection: Represents relationships between users (e.g., alumni and students). -

Feedback: Captures user feedback on jobs, events, and mentorship experiences. ####

Relationships: -

- ✔ Users can create and manage profiles. -
- ✔ Users can post jobs and events. -
- ✔ Users can connect with each other. -
- ✔ Users can provide feedback on jobs and events.

Functional Overview: -

Registration/Login: Users register and log in to access the platform. -

Profile Management: Users create and edit their profiles. -

Job Management: Users can post, browse, and apply for jobs. -

Event Management: Users can create and RSVP to events. -

Networking: Users can connect and communicate with each other.

2. Logical Model

The logical model translates the conceptual model into a more detailed design, defining the structure of the data, the relationships between entities, and the constraints without specifying how they will be implemented.

Entity-Relationship Diagram (ERD):

The logical model can be represented using an ERD, which includes: -

Entities: -

User: Attributes (user_id, name, email, password, role). -

Profile: Attributes (profile_id, user_id, bio, skills). -

Job: Attributes (job_id, title, description, company_id, posted_date). -

Event: Attributes (event_id, title, date, location, description). -

Connection: Attributes (connection_id, user_id_1, user_id_2). -

Feedback: Attributes (feedback_id, user_id, job_id, rating, comment). -

Relationships:

A **User** can have one or more **Profiles** (1-to-1). -

A **User** can post multiple **Jobs** (1-to-many). -

A **User** can create multiple **Events** (1-to-many). -

A **Job** can receive multiple **Feedback** entries (1-to-many). -

A **User** can connect with multiple other **Users** through **Connections** (many-to-many).

Logical Data Structure: -

User Table: - Fields: user_id (PK), name, email, password, role, status. -

Profile Table: - Fields: profile_id (PK), user_id (FK), bio, skills. -

Job Table: - Fields: job_id (PK), title, description, company_id (FK), posted_date. -

Event Table: - Fields: event_id (PK), title, date, location, description, creator_id (FK). -

Connection Table: - Fields: connection_id (PK), user_id_1 (FK), user_id_2 (FK). -

Feedback Table: - Fields: feedback_id (PK), user_id (FK), job_id (FK), rating, comment.

3. Physical Model

The physical model specifies how the logical model will be implemented in a specific database management system (DBMS). It includes details about the database schema, data types, indexing, and storage.

Database Schema:

The physical model defines the actual tables, columns, and data types used in the database.

User Table:

- user_id: INT, AUTO_INCREMENT, PRIMARY KEY -

name: VARCHAR(100)

- email: VARCHAR(100), UNIQUE

- password: VARCHAR(255) -

role: ENUM('admin', 'alumni', 'student', 'faculty', 'company')

- status: ENUM('active', 'pending', 'suspended')

Profile Table:

- profile_id: INT, AUTO_INCREMENT, PRIMARY KEY

- user_id: INT, FOREIGN KEY (references User(user_id))

- bio: TEXT - skills: VARCHAR(255) -

Job Table:

- job_id: INT, AUTO_INCREMENT, PRIMARY KEY

- title: VARCHAR(100)

- description: TEXT - company_id: INT, FOREIGN KEY (references User(user_id))

- posted_date: DATETIME

Event Table:

- event_id: INT, AUTO_INCREMENT, PRIMARY KEY
- title: VARCHAR(100)
- date: DATETIME
- location: VARCHAR(255)
- description: TEXT
- creator_id: INT, FOREIGN KEY (references User(user_id))

Connection Table:

- connection_id: INT, AUTO_INCREMENT, PRIMARY KEY
- user_id_1: INT, FOREIGN KEY (references User(user_id))
- user_id_2: INT, FOREIGN KEY (references User(user_id))

Feedback Table:

- feedback_id: INT, AUTO_INCREMENT, PRIMARY KEY
- user_id: INT, FOREIGN KEY (references User(user_id))
- job_id: INT, FOREIGN KEY (references Job(job_id))
- rating: INT CHECK (rating BETWEEN 1 AND 5)
- comment: TEXT

3.4.1 Scenarios

Scenario 1: Alumni Registration and Verification Actors:

Alumni, Admin

Steps:

1. **Registration**: An alumnus visits the ASTU Almunilink platform and fills out the registration form, providing their name, email, graduation year, and professional details.
2. **Pending Status**: Upon submission, the alumnus's account is marked as "pending" until verified by the Admin.
3. **Admin Review**: The Admin receives a notification of a new registration and accesses the verification queue.
4. **Verification Process**: The Admin reviews the submitted information, cross-references it with university records, and finds it valid.
5. **Approval:** The Admin approves the registration, changing the status to "active" and sending a confirmation email to the alumnus.

6. **Access Granted:** The alumnus can now log in and access features such as networking and job postings.

Scenario 2: Student Searching for Job Opportunities Actors:

Student, Alumni

Steps:

1. **Login**: A current student logs into their account on the ASTU Almunilink platform.
2. **Dashboard Navigation**: The student navigates to the "Job Listings" section from the dashboard.
3. **Job Search:** The student uses filters (e.g., industry, job type) to search for relevant job postings.
4. **Application:** The student finds a job that interests them and clicks to view the details. They prepare their application materials and submit their application through the platform.
5. **Confirmation**: The system confirms receipt of the application, and the student receives a notification of their application status.

Scenario 3: Faculty Posting Research Opportunities Actors:

Faculty, Admin

Steps:

1. **Login:** A faculty member logs into the system and accesses their dashboard.
2. **Create Opportunity:** The faculty member navigates to the “Research Opportunities” section and fills out a form detailing the project, requirements, and application process.
3. **Submit for Approval:** The faculty submits the opportunity, which is sent to the Admin for review.
4. **Admin Review:** The Admin receives the submission and reviews the opportunity for appropriateness and relevance.
5. **Approval:** The Admin approves the posting, making it visible to students and alumni.
6. Notification: The faculty member receives a notification that their opportunity has been published.

Scenario 4: Company Posting Job Openings Actors:

Company, Alumni, Students

Steps:

1. **Login**: A representative from a company logs into their account.
2. **Post Job**: The company navigates to the "Job Listings" section and fills out a form to post a new job opening, including details about the position and application process.
3. **Job Review**: After submitting, the job posting is sent for Admin review.

4. Admin Approval: The Admin reviews the job posting for compliance with platform standards and approves it.
5. **Visibility:** The job is now visible to all users (students and alumni).
6. **Applications:** Students and alumni apply for the position directly through the platform, and the company receives notifications of new applications.

Scenario 5: Networking Between Alumni and Students Actors:
Alumni, Students
Steps:
1. **Login**: An alumnus logs into their account and navigates to the networking section.
2. **Search Connections**: The alumnus uses the search functionality to find current students in their field of expertise.
3. **Send Connection Request:** The alumnus sends connection requests to a few students, indicating their interest in providing mentorship.
4. **Student Response:** The students receive connection requests, review the alumnus profiles, and accept the requests.
5. **Messaging**: After connecting, the alumnus and students can communicate via the platform's messaging system to discuss career advice and opportunities.

Scenario 6: Admin Managing User AccountsActors:
Admin
Steps:
1. **Login**: The Admin logs into their dashboard.
2. **User Management**: The Admin navigates to the "User Management" section to review a list of all registered users.
3. **Handle Pending Verifications**: The Admin finds several accounts pending verification and reviews each profile.
4. **Approve/Deny Requests**: Based on the information, the Admin approves some accounts and denies others, providing reasons for denial.
5. **Monitor Activity**: The Admin accesses activity logs to monitor user engagement and identify any suspicious activity.
6. **Data Reports**: The Admin generates a report on user demographics and job postings for administrative meetings.

Scenario 7: Event Management Actors:

Alumni, Faculty, Admin

Steps:

1. **Create Event**: An alumnus or faculty member logs in and creates a networking event, filling in details such as date, time, location, and description.
2. **Submit for Approval**: The event is submitted for Admin approval.
3. **Admin Review**: The Admin reviews the event details to ensure they meet platform guidelines.
4. **Event Approval**: The Admin approves the event, and it becomes visible on the platform's event calendar.
5. **User RSVP**: Users (students, alumni) can view and RSVP to the event, and the event organizer receives notifications of attendees.
6. **Event Reminder**: As the event date approaches, the system sends reminders to all registered participants.

3.4.2 Use Case Model

This diagram illustrates the interactions between users and the system:

Actors:

- Alumni

- Students

- Faculty

– Companies

- Admin

Use Cases:

- Register/Login

- Create/Edit Profile

- Post Job

- Apply for Job

- Create/Event

- Send Message

- Provide Feedback

Use Case descrition

Key Use Cases

1. User Registration - Actors: Alumni, Students, Faculty, Companies - Preconditions: User is on the registration page. - Postconditions: User account is created, and a confirmation email is sent. - Main Flow: 1.
User fills in the registration form with required information (name, email, password, role).
2. User submits the form.
3. System validates the input.
4. System creates a new user account in the database.
5. System sends a confirmation email to the user.

2. User Login - Actors: Alumni, Students, Faculty, Companies - Preconditions: User has a registered account. - Postconditions: User is logged in and directed to their dashboard.
- Main Flow:
1. User enters their email and password on the login page.
2. User submits the login form.
3. System validates the credentials.
4. If valid, the system logs the user in and redirects them to their dashboard.
5. If invalid, the system displays an error message.

3. Profile Management - Actors: Alumni, Students, Faculty, Companies - Preconditions: User is logged in. - Postconditions: User profile is updated in the database.
- Main Flow:
1. User navigates to the profile management section.
2. User edits their profile information (bio, skills, work experience).

3. User submits the changes.
4. System validates the input.
5. System updates the user profile in the database.

4. Job Posting - Actors: Alumni, Companies - Preconditions: User is logged in and has the role of Alumni or Company. - Postconditions: Job posting is created and submitted for approval.
- Main Flow:
1. User navigates to the job posting section.
2. User fills in the job details (title, description, requirements).
3. User submits the job posting.
4. System stores the job details and marks it as pending approval.

5. Job Application - Actors: Students, Alumni - Preconditions: User is logged in and has found a job listing. - Postconditions: Application is submitted and stored in the database.
- Main Flow:
1. User views job listings and selects a job.
2. User clicks on “Apply” and fills in the application form.
3. User submits the application.
4. System validates the application and stores it in the database.

6. Event Creation - Actors: Alumni, Faculty - Preconditions: User is logged in. - Postconditions: Event is created and submitted for approval.
- Main Flow:
1. User navigates to the event creation section.
2. User fills in event details (title, date, location, description).
3. User submits the event for approval.
4. System stores the event details and marks it as pending approval.

7. Networking (Connection Requests) - Actors: Alumni, Students - Preconditions: User is logged in. - Postconditions: Connection request is sent or accepted.
- Main Flow:
1. User searches for other users to connect with.

2. User sends a connection request.

3. If the other user accepts, the connection is established.

4. System updates the connection status in the database.

8. Admin User Management - Actors: Admin - Preconditions: Admin is logged in. - Postconditions: User accounts are verified, approved, or suspended.

- Main Flow:

1. Admin navigates to the user management section.

2. Admin reviews pending user registrations.

3. Admin approves or denies user accounts, providing reasons for denial if applicable.

4. System updates the user status in the database.

9. Admin Job Management - Actors: Admin - Preconditions: Admin is logged in. - Postconditions: Job postings are approved or removed. -

Main Flow:

1. Admin navigates to the job management section.

2. Admin reviews pending job postings.

3. Admin approves or denies job postings, providing reasons if applicable.

4. System updates the job status in the database.

10. Admin Event Management - Actors: Admin - Preconditions: Admin is logged in. - Postconditions: Events are approved, denied, or removed.

- Main Flow:

1. Admin navigates to the event management section.

2. Admin reviews pending events.

3. Admin approves or denies events, providing reasons if applicable.

4. System updates the event status in the database.

Use Case Diagram