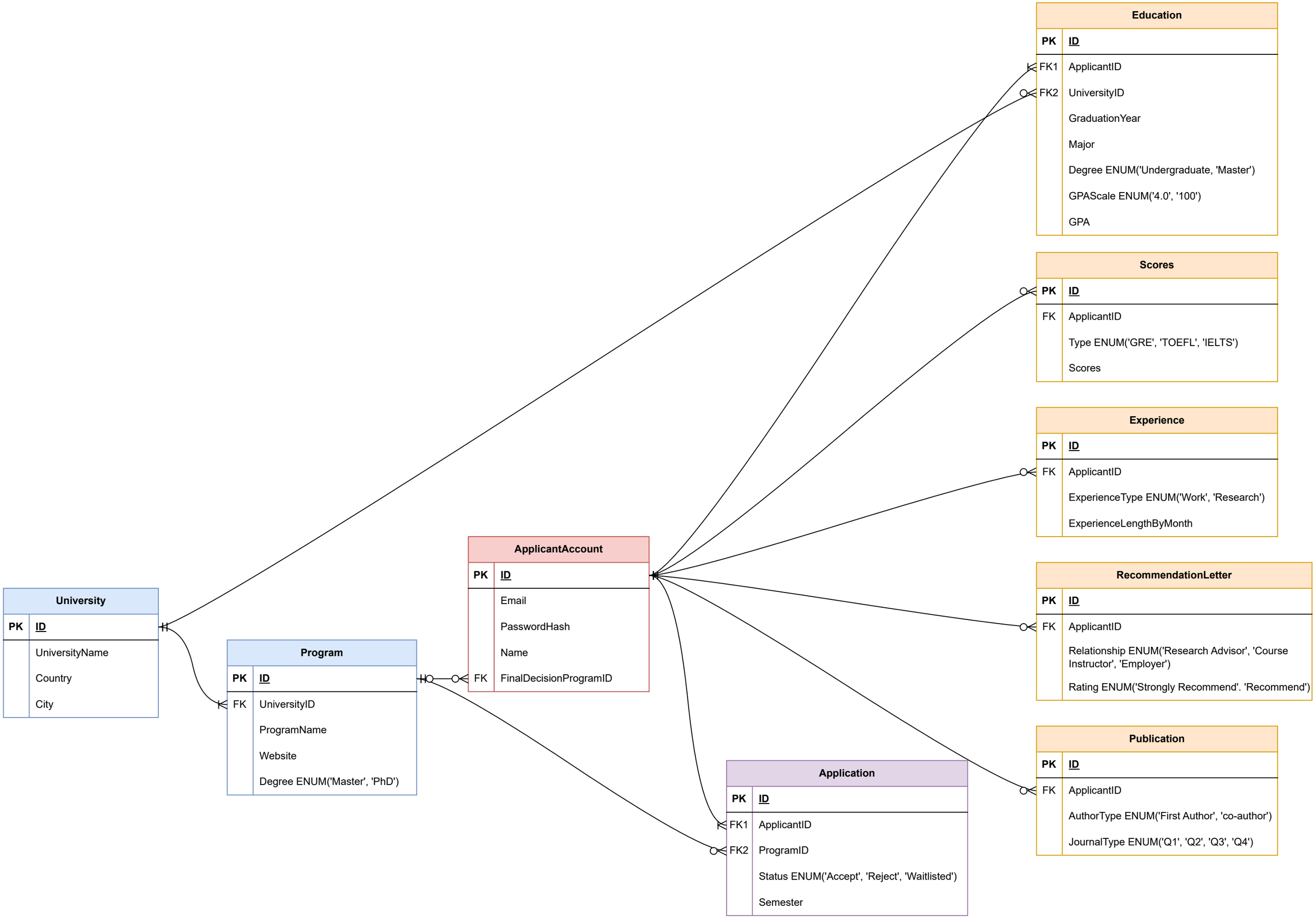

Education
PK
ID
FK1
ApplicantID
FK2
UniversityID
GraduationYear
Major
Degree ENUM('Undergraduate, 'Master')
GPAScale ENUM('4.0', '100')
GPA
Scores
PK
ID
FK
ApplicantID
Type ENUM('GRE', 'TOEFL', 'IELTS')
Scores
Experience
PK
ID
FK
ApplicantID
ExperienceType ENUM('Work', 'Research')
ExperienceLengthByMonth
RecommendationLetter
PK
ID
FK
ApplicantID
Relationship ENUM('Research Advisor', 'Course Instructor', 'Employer')
Rating ENUM('Strongly Recommend'. 'Recommend')
Publication
PK
ID
FK
ApplicantID
AuthorType ENUM('First Author', 'co-author')
JournalType ENUM('Q1', 'Q2', 'Q3', 'Q4')
ApplicantAccount
PK
ID
Email
PasswordHash
Name
FK
FinalDecisionProgramID
University
PK
ID
UniversityName
Country
City
Program
PK
ID
FK
UniversityID
ProgramName
Website
Degree ENUM('Master', 'PhD')
Application
PK
ID
FK1
ApplicantID
FK2
ProgramID
Status ENUM('Accept', 'Reject', 'Waitlisted')
Semester

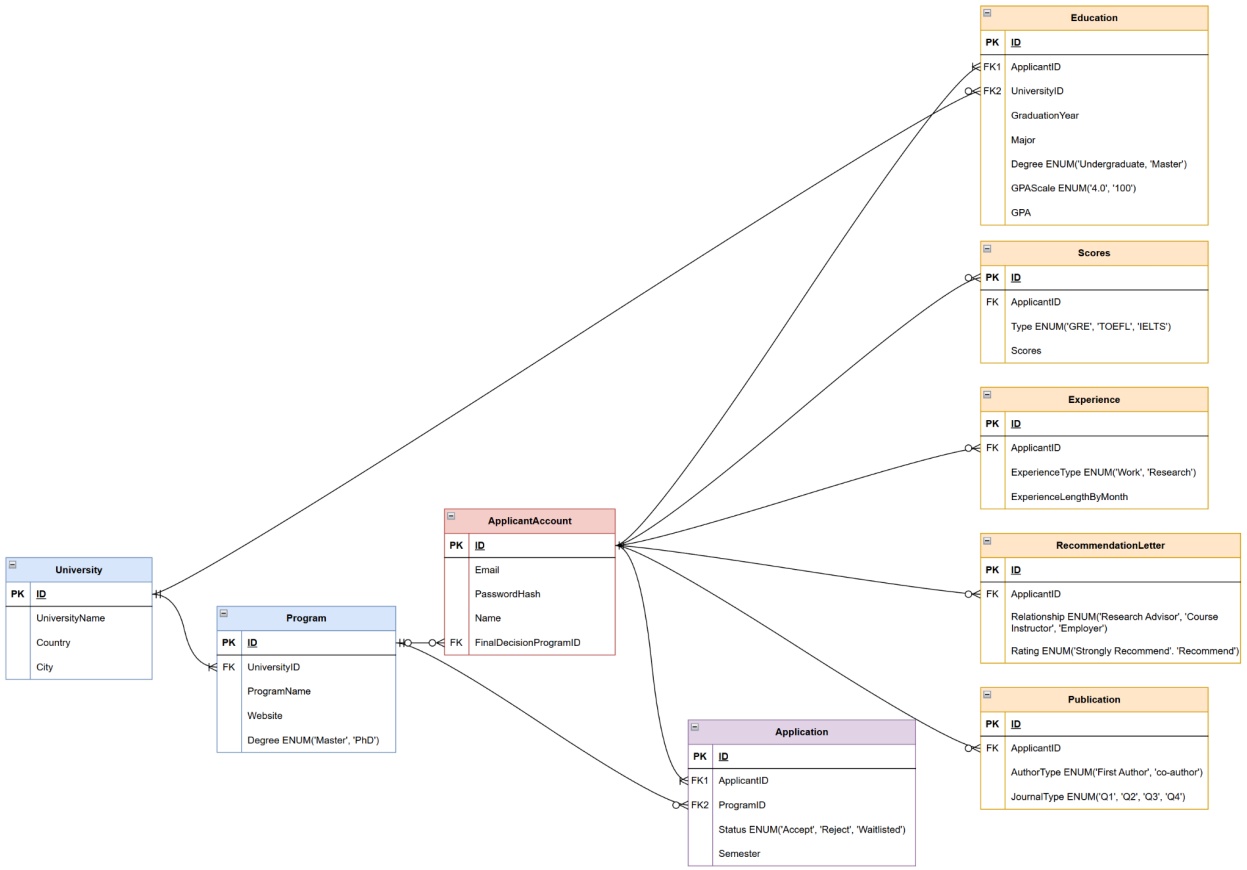
CS480 Project - CS Grad Application

Group Members:
Kening Zheng 679945030
Xia Wang 677945924
Huanhuan Ma 670964530
Xiaoqi Liu 654213498

Project Description:
We want to build an online website which summarizes the graduate universities applications from across the world, every one can submit their application result including which university accepted or rejected them along with their academic background. So that the new applicant could review the database and compare with their own background and get to know which universities they have a good chance to get the offer.

The database we will use at first is: https://opencs.app/. This website summarizes the application results for Chinese students to applying US graduate schools.
In the future, we may add more international database from https://www.thegradcafe.com/

ER-Diagram



Please see the link below or the attached pdf file for a more clear ER-Diagram
https://drive.google.com/file/d/1j9d64_3OYWofEO3JI10uIHtO507SoMJn/view?usp=sharing

Diagram Description:
We have 9 tables: University, Program, ApplicantAccount, Education, Scores, Experience, RecommendationLetter, Publication, and Application.

User Manual:
Firstly, the server will upload a list of universities from China, US, UK, etc, into University, then for each university, the server will also upload the programs related to computer science into Program.

Then, if a user/applicant want to use this website, they need to create an account on ApplicantAccount with their Email, PasswordHash, Name/Nickname. After registration, they are required to add at least one Education with UniversityID references from University table (this step is mandatory), if they can't find their UniversityID, they can add a new university in the University table. After having an account, they can also add more details for their academic background including Scores(GRE, TOEFL, or IELTS), Experience(Work or Research), RecommendationLetter(Strongly Recommend or Recommend), and Publication(First Author or co-author) (this step is optional).

Finally, the user can add their application result into Application table with ProgramID references from Program table. If they couldn't find the program, similarly, they can add it in the Program table and the corresponding university in the University table.

Relationship Analysis

1. **University** and **Program** (1:M)
One University can have many Programs (at least one). One Program belongs to only one University.

2. **ApplicantAccount** and **Education** (1:M)
One ApplicantAccount can have many Education records (at least one). One Education record belongs to only one ApplicantAccount.

3. **University** and **Education** (1:M)
One University can be referenced in many Education records (can be zero). One Education record references only one University.

4. **ApplicantAccount** and **Scores** (1:M)
One ApplicantAccount can have many Scores records (can be zero). One Scores record belongs to only one ApplicantAccount.

5. **ApplicantAccount** and **Experience** (1:M)
One ApplicantAccount can have many Experience records (can be zero). One Experience record belongs to only one ApplicantAccount.

6. **ApplicantAccount** and **RecommendationLetter** (1:M)
One ApplicantAccount can have many RecommendationLetter records (can be zero). One RecommendationLetter record belongs to only one ApplicantAccount.

7. **ApplicantAccount** and **Publication** (1:M)
One ApplicantAccount can have many Publication records (can be zero). One Publication record belongs to only one ApplicantAccount.
(It could be M:N for authors and publications, but here we only consider the number of publications an application has.)

8. **ApplicantAccount** and **Application** (1:M)
One ApplicantAccount can have many Application records (at least one). One Application record belongs to only one ApplicantAccount.

9. **ApplicantAccount** and **Program** (M:N) - (via Application table)
One ApplicantAccount can have many applications, which can be associated with many Programs.
One Program can be associated with many applications from many ApplicantAccounts.

10. **ApplicantAccount** and **Program** (1:M) - (via FinalDecisionProgramID)
This is a special relationship. One Program can be chosen as the final decision by many ApplicantAccounts (can be zero). One ApplicantAccount can choose one or zero (before the decision deadline) Program as his/her final decision.

List of Web Technologies

Frontend

- Framework and Language: Use React as the primary frontend framework to build a responsive and SEO-friendly user interface. TypeScript will ensure type safety and maintainability of the codebase.
- UI and Styling: Adopt Tailwind CSS or Material UI to provide a clean, responsive, and consistent user experience across devices. Include a dark mode option and ensure accessibility (a11y) compliance with WAI-ARIA standards.
- Form Handling and Validation: Implement a multi-step form using React Hook Form with schema validation via Zod or Yup. Support dynamic field validation for GPA, standardized test scores, and degree details.
- State Management and Data Fetching: Integrate React Query or SWR for efficient data fetching, caching, and revalidation. Use URL-based search filters for user-friendly data exploration.
- Visualization and Interaction: Use Chart.js or ECharts for interactive visualization of admission statistics and trends. Provide filtering options to compare applicants with similar backgrounds.
- Internationalization and SEO: Integrate i18next for multi-language support and implement SEO optimizations such as metadata, sitemaps, and canonical URLs to increase discoverability.

Backend

- Framework and Runtime: The backend will be implemented using Node.js (NestJS/Express) or Python (FastAPI/Django REST Framework). It will expose RESTful or GraphQL APIs for frontend communication.
- Authentication and Authorization: Support both email-password login. Use JWT or secure cookies for session management. Implement Role-Based Access Control (RBAC) to distinguish between user, reviewer, and admin roles.
- Business Logic and API Design: Develop core APIs for university, program, and applicant data management. Include endpoints for submission review, data statistics, and trend analysis.
- Validation and Rate Limiting: Use schema validation (Zod or Pydantic) for input consistency. Implement API rate limiting and request throttling to prevent abuse and ensure system stability.

Database and Data Management

- Primary Database: Adopt MySQL as the main relational database for structured data storage. Design normalized tables to store applicants, education, scores, publications, and applications.
- ORM and Migration: Use ORM frameworks for model abstraction and schema migration management.
- Search and Indexing: Support complex queries with indexes and filtering on GPA, test scores, program type, and country. Introduce Elasticsearch or OpenSearch for advanced search and faceted filtering in later phases.
- Caching and Session Management: Utilize Redis for caching frequently accessed data, maintaining session tokens, and managing background tasks.
- Data Integration: Build scheduled ETL pipelines or API-based synchronization with data sources like opencs.app and TheGradCafe.

Infrastructure and Deployment

- Containerization and Deployment: Containerize services using Docker and manage deployments with Kubernetes or Docker Compose for scalability and fault tolerance.
- CI/CD Pipeline: Automate testing, linting, and deployment using GitHub Actions or GitLab CI/CD to ensure smooth integration and continuous delivery.
- Cloud and Storage: Deploy on AWS, GCP, or Azure using managed databases and storage services like Amazon S3 for backups and user-uploaded files.
- Monitoring and Logging: Use Prometheus and Grafana for real-time monitoring. Centralize logs using Elastic Stack (ELK) or similar for troubleshooting and analytics.

Security and Privacy

- Data Protection: Enforce HTTPS/TLS for all network communications and encrypt sensitive data at rest and in transit.
- Access Control: Limit access through least-privilege principles and audit sensitive operations to ensure accountability.
- Privacy Compliance: Ensure compliance with international data protection regulations (e.g., GDPR). Support data anonymization and user-initiated data deletion.
- Anti-Abuse Measures: Integrate CAPTCHA or Turnstile, request throttling, and input validation to defend against spam and automated attacks.

Github link: