Optical Marking Management System Project Proposal

Team 7 - Paula, Ishika, Bennett, Francisco, Dima

Project Purpose and Unique Value Proposition

- The purpose of this project is to provide educators with a versatile tool for creating, administering, and grading scanned bubble-sheet exams.

- Fast, efficient and will reduce human errors.
- Customizable bubble-sheet exams
- Processes existing formats

Success Criteria & Outcomes

- Project completed on time
- Project meets the requirements
- All planned features are satisfied by the deliverables
- Automated and manual tests are passing
- Stakeholder satisfied with delivered project

Project Scope to be completed by July 19th

Secure authentication

- OAuth and password-based authentication with encryption
- Automated tests to ensure API security

Efficient Exam Management

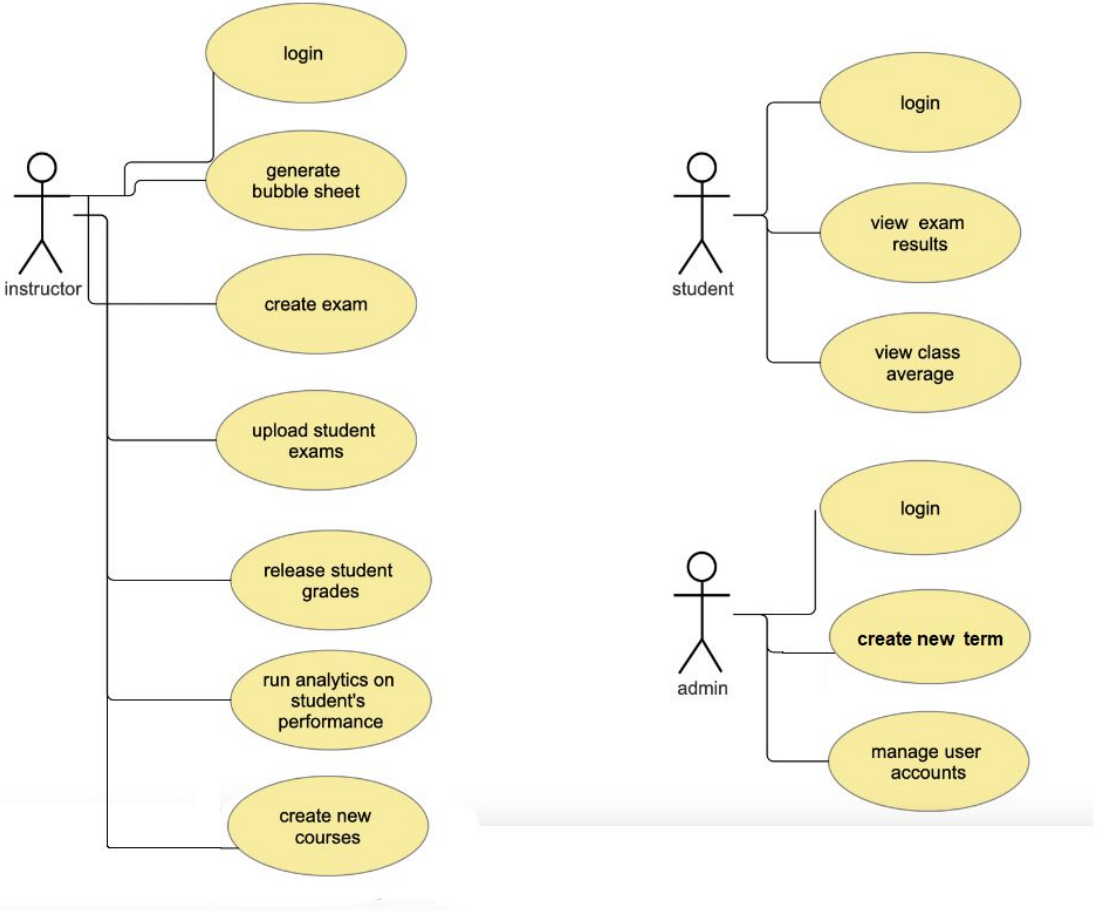
- User-friendly interface with common templates
- Less than 10 minutes to set up an exam
- Minimal UI with helpful tooltips

Student Access

- View scanned exam images online to verify grading

Performance evaluation

- Detailed analytics and reporting tools
- Year-over-year performance comparisons



USER GROUPS AND USE CASE DIAGRAM

Requirements

Functional Requirements

- Authentication and Joining Courses
- Exam Creation
- Exam Grading
- Analytics
- Feedback
- Administrative tasks

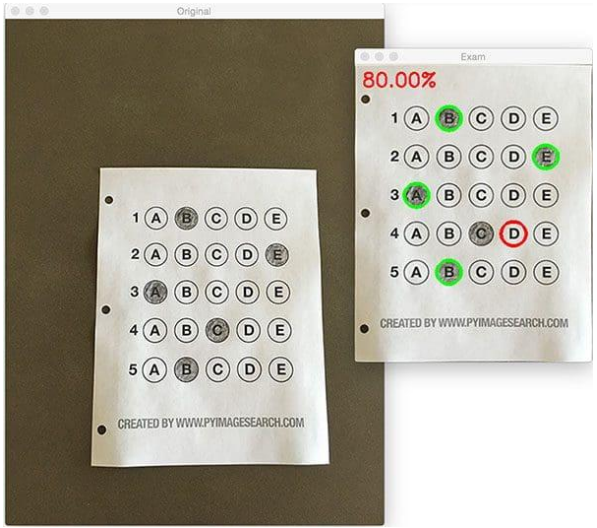


Image source: https://pyimagesearch.com/2016/10/03/bubble-sheet-multiple-choice-scanner-and-test-grader-using-omr-python-and-opencv/

User Requirements

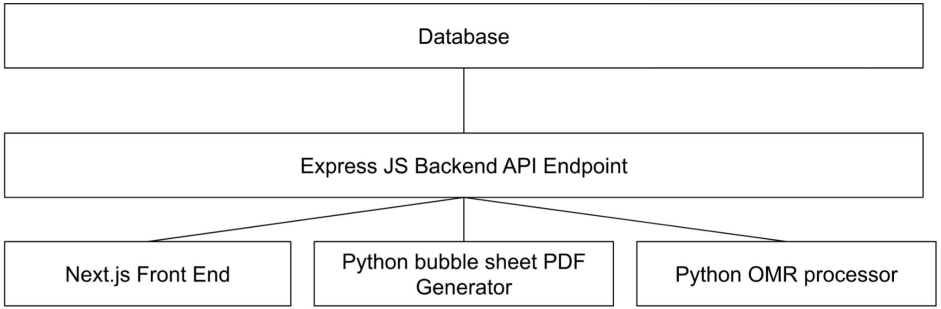
- Responsive and intuitive web application for all users
- Instructors
  - Create/load a bubble-sheet exam
  - Create classes and exams
  - Receive automatically-generated grades and control their release
  - Visualise student performance
    - Metrics for individual course instances
    - Year-over-year performance analysis
  - Download exam results
- Students
  - Receive feedback on their performance for an active course
- Administrators
  - Maintain and manage the system

Technical Requirements

- **Maintainability**: Use of a microservices architecture
  - Communication happens through API endpoints
  - Apache/Nginx proxies
  - ExpressJS middleware for database communication
- **Security**:
  - Encryption
  - Use of an external authentication provider to limit information shared
- **Optical Mark Recognition (OMR) system**:
  - Use of an open-source pre-trained OMR model
    - No GPU availability will be assumed
  - Output in JSON format
- **Database**:
  - Use of a relational database (PostgreSQL)

Technical Requirements

- **Front-end**: NextJS
- **Back-end**: ExpressJS
- **CI/CD**: GitHub Actions
- **Testing**: Unit Testing, Integration testing, End-to-end testing
    - Jest (Javascript), unittest (Python), and Cypress (Nextjs)
- **Containerization**: Docker
- **Documentation**
    - Components and functions will be documented with JSDoc (Next.js) and Docstrings (Python)
    - Detailed Pull Requests
    - README for each microservice



Nice-to-haves

All Users:

- System onboarding

Instructors:

- Changing the weighting of questions in an exam
- Upload a CSV file of students for student registration
- Up to 500 students per course
- Support of multiple versions of the same exam
- Multiple answers per question

Administrators:

- Monitoring system performance and usage

Students:

- Detailed performance metrics for individual students