Insight Glass: Project Presentation

Software Engineering Fundamentals, CIE 460

Aser Osama 202101266 Aya Sherif 202100642
Gehan Sherif 201902069 Omar Ayman 202100443

May 19, 2024

Project Scope and Requirements

Project Scope

- Develop a one of a kind in Egypt web app for professionals.
- Help professionals from different disciplines start their careers at companies where they can thrive.
- Provide a platform for companies to find the best candidates for their job.

Functional Requirements

- User Registration and Authentication
  - Create accounts with email, username, and password.
  - Email OTP authentication for security.
- Profile Management
  - Maintain personal profiles with skills, experience, and education.
  - Upload and manage resumes and documents.

Functional Requirements (cont.)

- Advanced Company & Job Search Filters
  - Search jobs by keyword, location, industry, or company.
  - Filter jobs by salary, type, and experience level.
- Comprehensive Company Profiles
  - Access detailed company profiles with key details.
  - View employee testimonials for workplace insights.

Functional Requirements (cont.)

- Transparent Job Listings
  - Access detailed job listings which include responsibilities and benefits.
  - Track application status and updates.
  - Access company reviews and ratings.
- Career Advice
  - Access articles, tips, and resources for career development.
  - Participate in forums and Q&A sections.

Functional Requirements (cont.)

- Interactive Discussion Forums
  - Join specialized forums for job discussions.
  - Engage with industry experts and mentors.
- Salary Benchmarking Tools
  - Access salary data and compare against peers.
  - View salary visualizations like charts and graphs.

Functional Requirements (cont.)

- Feedback and Ratings System
  - Rate companies and job listings on a standardized scale.
  - Provide anonymous feedback.
- Integration with Social Media Platforms
  - Share job listings and profiles on social media.
  - Link platform profiles with social media accounts.

Functional Requirements (cont.)

- Interview Preparation Resources
  - Share and access interview experiences.
  - Use guides for various interview types.
- Analytics and Insights Dashboard
  - Admin access to track user engagement and trends.
  - Comprehensive data visualization tools.

Non-functional Requirements

- Security and Privacy (SSL Encryption, Identity Framework, etc.)
- Performance (Caching & Load Balancing)
- Scalability (Azure Scalable Web Apps)
- Reliability (Constant Monitoring, Logging, Alerts & Testing)

Project Design and Architecture

Project Architecture

We followed a 3-tier architecture with all components hosted as scalable components on Azure.

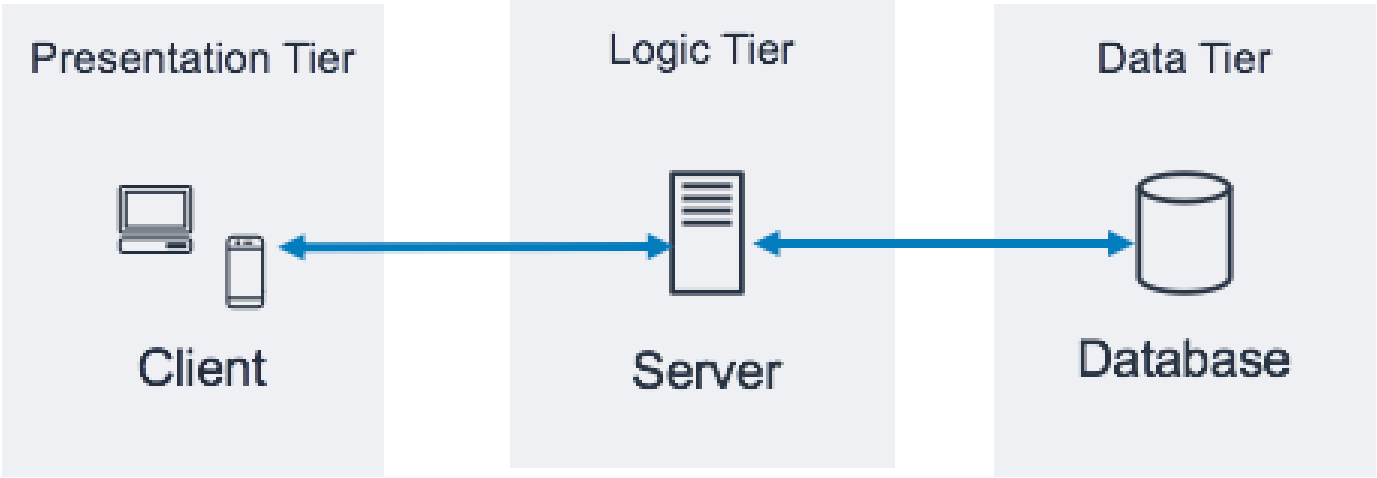

Figure 1: Archircture Diagram

Project Archircture (cont.)

- Frontend & Backend: React.js & ASP web API hosted on Azure Web Apps as a containerized scalable app.
- Database: Azure MySQL Database for storing user, job, and company data.
- Devops: CI/CD Pipeline using GitHub Actions for automated deployment.

Design Patterns and OO Principles

Design Patterns:

- Singleton Pattern for React Context: Helps us persist user state across components.
- Factory Pattern for the database connection: Make the most out of Entity Framework caching and performance by following it's best practices.
- Dependency Injection in Web API controllers: We inject services and database context to controllers.
- Dependency Injection in React: We build our components to be reusable.

Design Patterns and OO Principles (cont.)

SOLID Principles

- Single Responsibility Principle: Each of the controllers is responsible for a single model.
- Open/Closed Principle: API controllers are easily extendable.
- Interface Segregation Principle: We can use partial classes in C#.
- Dependency Inversion Principle: All communication was top down.

Devops

Scrum Implementation

- We used GitHub Projects to manage our Scrum workflow.
- We help bi-weekly stand-ups that were changed to bi-daily in later stages.
- We had multiple Sprint Reviews and Retrospectives

Scrum Implementation (cont.)

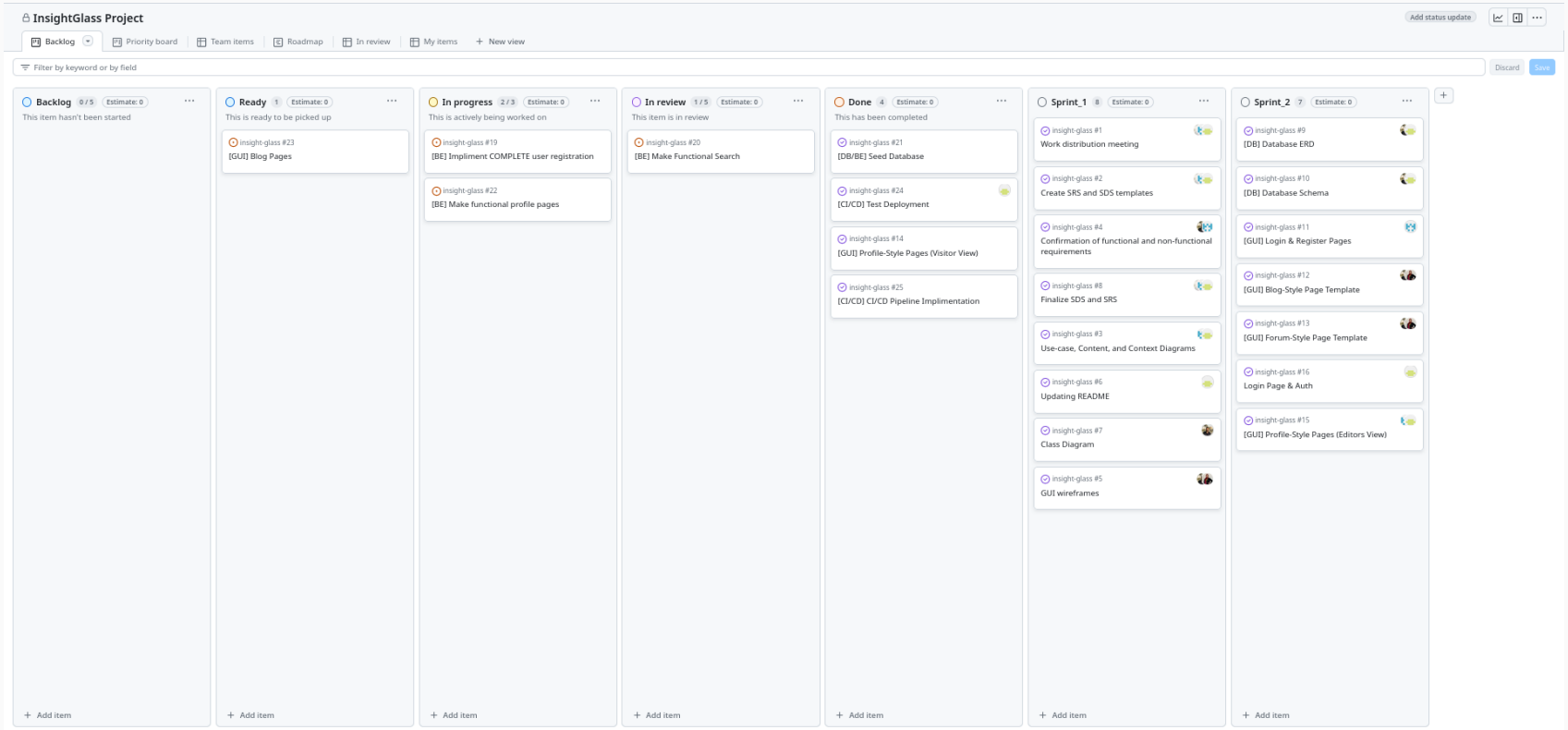

Figure 2: Scrum Board

CI/CD Pipeline

1. Version Control using Git and GitHub
2. Automated Unit Testing using XUnit, Moq, & Entity Framework Core - InMemory,
3. Automated Integration Testing using Selenium (in progress)
4. Continuous Integration using GitHub Actions to Azure Container Registry
5. Continuous Deployment to Azure Web App

CI/CD Pipeline

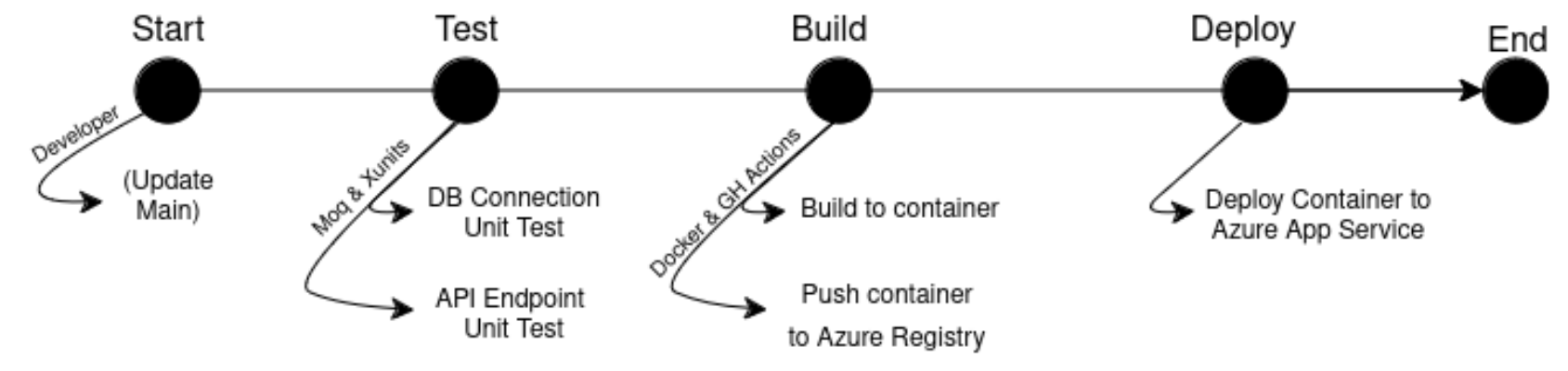

Figure 3: CI/CD Pipeline

Testing

Unit Tests

- We used XUnit for unit testing.
- We used Moq for mocking controllers.
- We used Entity Framework Core - InMemory for database testing.
- We used Selenium for integration testing. (in progress)

Tests

Some of the tests we implemented:

- Testing the connection to prod. database.
- Mock testing the “companies” API controller.
- Mock testing the “seekers” API controller.

Reflections

Challenges Encountered

- **Technical Challenges**
  - Learning React and .NET
  - Lack of resources (using latest releases of .NET and React.)
  - Lack of mentorship.
  - Time constraints due to semester schedule and holidays.

What Went Wrong

- **Time Management**
  - Overly optimistic estimates.
  - Delayed starts and rushed efforts due to clashing deadlines.
- **Task Breakdown**
  - Ineffective task division.
  - Underestimated complexity of React and .NET integration.
- **Documentation**
  - Lack of internal documentation (Common flaw of Scrum/Agile development).
  - Inconsistent coding standards initially.

What Went Well

- **Team Collaboration**
  - Effective collaboration and communication.
  - Peer support and knowledge sharing.
- **Adaptability**
  - Quickly learned and applied new technologies.
  - Improved technical skills and confidence.

How to Improve

- **Start Earlier**
  - Begin planning and development sooner.
- **Seek Mentorship**
  - Engage mentors early for feedback.
- **Better Time Management**
  - Implement realistic time estimates.
  - Use project management tools effectively.
- **Effective Task Breakdown**
  - Break tasks into smaller, manageable units.
  - Regularly review and adjust task allocations.

How to Improve (cont.)

- **Documentation**
  - Maintain comprehensive documentation.
- **Continuous Feedback**
  - Establish regular check-ins and feedback loops.

Conclusion

- Significant learning experience.
- Developed crucial skills and insights.
- Future improvements will enhance project success.

Discussion

Questions?