# JobQuest Navigator – Week 6 Update Report

Maria Soto, Shruti Amit Vasanwala, Zhihuai Wang, Ishan Aakash Patel

Team 9

The Zombies of CAA

Seneca Polytechnic

Course Code: CAA900

David Chan

## Project Summary

**Weekly Meeting Minutes and Materials**

Access the documents from our weekly meetings

[00 Documents/0099 Final Documents/Reports/Meetings](https://github.com/MariaVSoto/JobQuest_Navigator_CAA/tree/main/00%20Documents/0099%20Final%20Documents/Reports/Meetings)

## Accomplishments

## Completed Work

**Epic 1 and IU Design**

- Migrated project from Flask to Django, removed all legacy Flask code.

- Cleaned up directory, keeping only Django, frontend, requirements, and documentation.

- Verified Django project structure and API endpoints.

- Supplemented and aligned API fields with Adzuna CSV sample.

- Generated and updated OpenAPI (Swagger) documentation.

- Cleaned up Python cache, old scripts, and unnecessary files.

- Ensured frontend and backend field consistency.

- Automated and documented the cleanup process.

1. 404 Page Implementation

- Created a modern, user-friendly 404 page

- Implemented responsive design with SVG illustration

- Added fallback mechanism for image loading

- Integrated with React Router for automatic routing

2. Navigation Bar Enhancement

- Reorganized navigation items into logical groups:

\* Main Navigation (Jobs, Map, Dashboard)

\* Job Management (Saved Jobs, Application History)

\* Career Tools (Resume Builder, Interview Prep)

\* Account (Profile, Settings)

\* Authentication (Login, Sign Up)

- Improved UI/UX with:

\* Consistent spacing and alignment

\* Visual separators between groups

\* Better responsive behavior

\* Enhanced active state indicators

3. Code Quality Improvements

- Implemented component-based architecture

- Added proper error handling

- Improved code organization and maintainability

- Enhanced responsive design implementation

Next Steps:

1. Implement user authentication flow

2. Add loading states for async operations

3. Enhance error handling across components

4. Add unit tests for new components

**Epic 3**

- Epic 3 provides AI-driven suggestions for improving resumes and includes mechanisms for user feedback.

- It enhances user resumes with intelligent suggestions and continuously improves AI models based on feedback.

- The system consumes resume data from Epic 2 and can use job data from Epic 1 to generate tailored suggestions.

- Frontend components are designed to display suggestions, collect user feedback, and integrate with resume editing workflows.

**Epic 6**

- Created a new Django project and app (company\_research, api).

- Generated requirements.txt for Django REST API development.

- Created README.md with project overview, setup, and structure.

- Created a basic frontend skeleton (frontend/index.html).

- Created a Django test skeleton (tests/test\_api.py).

- Initialized Task Master for task management.

- Ensured all project scaffolding aligns with best practices.

## Challenges or Blockers

|  |  |  |
| --- | --- | --- |
| **User Story** | **Task** | **Challenges or Blockers** |
| 0 | UI/UX | Backend integration complexity |
| 0 | UI/UX | Browser compatibility |
| 0 | UI/UX | Navigation organization: Implemented logical grouping |
| 0 | UI/UX | Responsive design: Created flexible layouts |
| 3.1 | SM-20 | Ensuring alignment and seamless integration with Epic 1 and Epic 2 |
| 3.1 | SM-20 | Guaranteeing the accuracy and reliability of AI-driven suggestions |

## Plan for Next Week

Sprint: **Sprint 4**

**Sprint Dates:**08June 2025 to 15 June 2025

|  |  |  |
| --- | --- | --- |
| **Epic** | **Owner** | **Task** |
| 0 | Shruti / All Members | Integration design with Epics |
| 2 | Kevin | User Story 2.1: Build AI-driven resume version control system. |
| 3 | Maria | User Story 3.1 (B): Implement AI suggestions for resume alterations. |
| 4 | Ishan | User Story 4.2: Fetch real-time certification recommendations from Indeed. |
| 4 | Ishan | User Story 4.3: Design certification roadmap UI. |

## Sprint Status and Timeline

* **Progress Tracking:**
  + Use Jira for task management and sprint tracking.
    - [List - The Zombies of CAA - Jira](https://myseneca-team-pi6s3gm8.atlassian.net/jira/software/projects/SM/list?sortBy=customfield_10015&direction=ASC)
  + Update GitHub weekly for documentation and progress updates.
    - <https://github.com/MariaVSoto/JobQuest_Navigator_CAA/tree/dev>
* **Next Steps:**
  + Work on assigned tasks per the above plan.
  + Address blockers and share progress in weekly meetings.
  + Ensure all documentation and code are up-to-date in the project repository.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Tools to develop the project

Based on project requirements and your preferences, the following technology stack is selected:

* **Frontend:** React.js
* **Backend:** Django (Python, full-featured, powerful ORM, suitable for rapid development and robust microservices)
* **Backend runtime:** Lambda (serverless deployment for backend services; enables automatic scaling, pay-per-use billing, and simplified infrastructure management. Each microservice or API endpoint can be deployed as an independent Lambda function, integrated with API Gateway for HTTP access. This approach reduces operational overhead and is well-suited for event-driven or microservice architectures.)
* **Database:** MySQL (for all structured data, including users, jobs, applications, certifications, interviews, etc.)
* **File/Object Storage:** AWS S3 (for all resume files and version history, only metadata stored in MySQL)
* **Map/Geo Service:** Google Maps API (for geolocation display and geocoding)
* **AI Service:** OpenAI API (for AI-powered suggestions, interview preparation, etc.)
* **Job Data API:** Adzuna (the only job data aggregation source for now, extensible in the future)
* **CI/CD:** GitHub Actions (for automated testing, building, and deployment)
* **CD Environment & Infrastructure:** Provisioned and managed using Terraform
* **API Debugging & Testing:** Performed using Postman