**1. Planning and Requirements Gathering**

**a. Define Project Scope and Objectives**

* **Clarify Features:** Detail all features your OSV should have (e.g., interactive maps, search functionality, user roles).
* **Stakeholder Consultation:** Engage with stakeholders in the National Congress to gather requirements and expectations.
* **Establish Metrics:** Establish how you will measure the success of the application (e.g., user adoption rates, performance benchmarks).

**b. Create a Project Timeline**

* **Milestones:** Set key milestones for each phase of the project.
* **Deadlines:** Assign realistic deadlines to ensure timely progress.
* **Resource Allocation:** Determine the resources (time, team members, tools) required for each task.

**2. Design Phase**

**a. Database Schema Design**

* **Data Modeling:** Identify all entities (e.g., departments, roles, employees) and their relationships.
* **Normalization:** Ensure the database is normalized to reduce redundancy and improve integrity.
* **Scalability:** Design the schema to accommodate future growth and changes in organizational structure.

**b. API Design and Architecture**

* **Endpoint Specification:** Define all necessary API endpoints for data retrieval, creation, updating, and deletion.
* **Authentication & Authorization:** Implement secure methods to ensure only authorized users can access or modify data.
* **Documentation:** Create comprehensive API documentation for ease of integration and future maintenance.

**c. User Interface (UI) and User Experience (UX) Design**

* **Wireframes and Mockups:** Develop wireframes and mockups to visualize the layout and user flow.
* **Responsive Design:** Ensure the application is accessible on various devices and screen sizes.
* **Accessibility Standards:** Adhere to accessibility guidelines (e.g., WCAG) to make the app usable for all users.

**3. Development Phase**

**a. Front-End Development**

* **Technology Stack:** Choose modern front-end frameworks (e.g., React, Vue.js, Angular) to build an interactive UI.
* **Component Development:** Create reusable UI components for consistency and efficiency.
* **State Management:** Implement state management solutions (e.g., Redux, Vuex) to handle application state effectively.

**b. Back-End Development**

* **Server Setup:** Set up the server environment using suitable technologies (e.g., Node.js, Django, Ruby on Rails).
* **API Implementation:** Develop the API endpoints as per the design specifications.
* **Database Integration:** Connect the back-end to the database, ensuring secure and efficient data handling.

**c. Automated Data Extraction Tools**

* **Tool Selection:** Choose appropriate tools or libraries for data extraction (e.g., web scraping frameworks, ETL tools).
* **Script Development:** Write scripts to automate the extraction, transformation, and loading (ETL) of organizational data.
* **Scheduling and Automation:** Implement scheduling (e.g., cron jobs) to automate regular data updates.

**4. Integration Phase**

**a. Front-End and Back-End Integration**

* **API Consumption:** Connect front-end components to back-end APIs for dynamic data rendering.
* **Error Handling:** Implement robust error handling to manage API failures gracefully.
* **Performance Optimization:** Optimize API calls to reduce latency and improve user experience.

**b. Database and Data Extraction Integration**

* **Data Validation:** Ensure that extracted data conforms to the database schema and business rules.
* **Automated Updates:** Set up processes to automatically update the database with new or modified data.

**5. Testing Phase**

**a. Unit Testing**

* **Front-End Testing:** Test individual UI components for functionality and reliability.
* **Back-End Testing:** Verify each API endpoint and back-end logic for correctness.

**b. Integration Testing**

* **End-to-End Testing:** Ensure that front-end and back-end systems work seamlessly together.
* **Data Flow Verification:** Confirm that data flows correctly from extraction tools through the database to the user interface.

**c. User Acceptance Testing (UAT)**

* **Stakeholder Involvement:** Allow stakeholders to test the application and provide feedback.
* **Iterative Improvements:** Make necessary adjustments based on UAT feedback to meet user expectations.

**d. Accessibility and Performance Testing**

* **Accessibility Compliance:** Use tools and manual testing to ensure the application meets accessibility standards.
* **Performance Benchmarks:** Test the application under various conditions to ensure it remains responsive and reliable.

**6. Deployment Phase**

**a. Prepare Deployment Environment**

* **Hosting Setup:** Choose and configure a hosting solution (e.g., cloud services like AWS, Azure, or on-premises servers).
* **CI/CD Pipeline:** Implement Continuous Integration and Continuous Deployment pipelines for automated testing and deployment.

**b. Deploy Application**

* **Staging Environment:** Deploy to a staging environment for final testing.
* **Production Deployment:** Move the application to the production environment, ensuring minimal downtime.

**c. Monitor and Maintain**

* **Monitoring Tools:** Set up monitoring to track application performance and uptime (e.g., New Relic, Grafana).
* **Error Tracking:** Implement tools to log and alert on errors (e.g., Sentry).
* **Regular Updates:** Schedule regular maintenance and updates to keep the application secure and up-to-date.

**7. Post-Deployment and Maintenance**

**a. User Training and Documentation**

* **User Guides:** Create comprehensive documentation and tutorials for end-users.
* **Training Sessions:** Conduct training sessions for stakeholders to familiarize them with the application.

**b. Feedback and Iterative Improvements**

* **Collect Feedback:** Continuously gather user feedback to identify areas for improvement.
* **Implement Enhancements:** Prioritize and implement enhancements based on user needs and feedback.

**c. Security and Compliance**

* **Regular Audits:** Perform regular security audits to protect sensitive organizational data.
* **Compliance Checks:** Ensure the application remains compliant with relevant laws and regulations.

**8. Prioritization of Tasks**

To effectively manage your workload, prioritize tasks based on their impact and dependencies:

1. **Foundational Setup:**
   * Define project scope and objectives.
   * Design the database schema.
   * Set up the development and deployment environments.
2. **Core Development:**
   * Develop back-end APIs.
   * Implement automated data extraction tools.
   * Develop front-end components and UI/UX design.
3. **Integration and Testing:**
   * Integrate front-end with back-end.
   * Conduct unit, integration, and user acceptance testing.
4. **Deployment and Monitoring:**
   * Deploy to staging and production environments.
   * Set up monitoring and error tracking.
5. **Finalization and Maintenance:**
   * Provide user training and documentation.
   * Gather feedback and implement iterative improvements.

**9. Tools and Technologies Recommendations**

* **Front-End:**
  + Frameworks: React, Vue.js, or Angular
  + Libraries: D3.js for interactive data visualization
* **Back-End:**
  + Frameworks: Node.js with Express, Django, or Ruby on Rails
  + Database: PostgreSQL or MySQL for relational data; consider NoSQL if needed
* **Data Extraction:**
  + Tools: Python with Beautiful Soup or Scrapy for web scraping; ETL tools like Apache NiFi
* **API Development:**
  + RESTful APIs or GraphQL depending on your data needs
* **Deployment:**
  + Platforms: AWS, Azure, or Google Cloud
  + CI/CD: Jenkins, GitHub Actions, or GitLab CI
* **Testing:**
  + Front-End: Jest, Mocha, or Cypress
  + Back-End: Postman for API testing, pytest for Python
* **Monitoring:**
  + Tools: New Relic, Grafana, Sentry

**10. Best Practices**

* **Version Control:** Use Git for source code management with a clear branching strategy.
* **Documentation:** Maintain thorough documentation for code, APIs, and user guides.
* **Agile Methodology:** Consider adopting Agile practices for iterative development and flexibility.
* **Regular Meetings:** Hold regular team meetings to track progress, address issues, and adjust plans as needed.
* **Security Best Practices:** Implement security measures such as data encryption, secure authentication, and regular vulnerability assessments.