JÖNKÖPING UNIVERSITY

Industrial placement course

Weekly report

Context
Patent Gap is an innovative legal technology startup company specializing in patent infringement detection and intellectual property management solutions. The company addresses a critical challenge in the intellectual property landscape: the time-consuming and labor-intensive process of identifying potential patent infringements through manual analysis. With the exponential growth of patent filings globally, patent attorneys and legal professionals require sophisticated technological solutions to efficiently monitor, analyze, and manage patent portfolios.

Role, Responsibilities and Tasks
The company, being fresh and focusing solely on one single product, maintains a lean organizational structure optimized for rapid development and innovation. My coworkers are mostly managerial staff, partners, attorneys and professionals from the legal background. I am responsible for the design, development, testing and deployment of the entire Patent Gap platform and I report directly to the owner Otto Werneskog who resides at Stockholm.

Week 1

Activities week 1

This week marked the initiation of the Patent Gap project with comprehensive groundwork. I began by conducting extensive research into patent infringement detection systems and analyzing existing solutions in the market. The primary focus was understanding the technical requirements for building a patent management platform. I designed the initial database architecture, determining the data structures needed for cases, patents, alerts, and user management. Significant time was dedicated to setting up the development environment, including configuring Flask as the backend framework and establishing the project repository structure. I also researched various patent data sources, particularly the USPTO API, to understand how to integrate external patent databases. The week concluded with creating preliminary wireframes for the frontend interface and documenting the technical specifications. This foundational work established clear objectives and a roadmap for the project's development phases.

Reflections week 1

Starting a new project always brings a mix of excitement and uncertainty. This week reinforced the importance of thorough planning before diving into implementation. I realized that understanding the patent domain requires significant domain-specific knowledge, which motivated me to research extensively. The challenge of designing a scalable database structure for patent data made me appreciate the complexity of information architecture. I learned that patent infringement detection isn't just about technical implementation – it requires understanding legal concepts, patent classification systems and similarity detection methodologies. The research phase was more time-consuming than anticipated, but it proved invaluable for making informed architectural decisions. I'm confident that the solid foundation laid this week will streamline development in subsequent phases. Moving forward, I need to maintain this balance between planning and execution.

Week 2

Activities week 2

Week two focused on establishing the core backend infrastructure. I implemented the Flask application structure with proper routing and initialized the database connectivity using MongoDB. The models directory was created with separate modules for cases, alerts, users, and demo requests, following a modular architecture pattern. I developed the authentication system with session management and created API endpoints for user login and logout. The database module was built to handle CRUD operations with functions for connecting to MongoDB, retrieving data, and performing updates. I also implemented environment variable management through the env_controller module, ensuring secure configuration handling across development and production environments. Initial API endpoints for case management were created, including retrieving user cases and case details. Testing was performed using Postman to verify endpoint functionality. Documentation began with the creation of README structure and technical specifications.

Reflections week 2

This week emphasized the importance of modular architecture in software development. Separating concerns into distinct modules (models, controllers, database) made the codebase more maintainable and scalable. I encountered challenges with MongoDB connection management, particularly handling connection pooling and error scenarios. Debugging authentication flows taught me valuable lessons about session management and security best practices. I learned that proper error handling from the start saves significant debugging time later. The decision to use environment variables for configuration proved wise as it simplified testing across different environments. Working with Flask deepened my understanding of web frameworks and RESTful API design principles. I recognized that investing time in proper project structure early pays dividends throughout the development. The experience reinforced that backend development requires careful attention to data consistency and security considerations.

Week 3

Activities week 3

This week centered on developing the patent processing pipeline and document handling capabilities. I implemented the data_processor module with functions for extracting text from PDF documents and generating embeddings. Research was conducted on various embedding techniques, leading to the implementation of both online (OpenAI API) and offline (TF-IDF) embedding generation methods. The file_controller module was created to handle document reading from URLs, supporting both PDF and XML formats. I developed keyword extraction functionality using natural language processing techniques. The patent similarity calculation system was implemented using cosine similarity between embedding vectors. Integration with the USPTO API began, requiring extensive documentation review and testing. Database schemas were refined to accommodate patent documents, embeddings and reference relationships. API endpoints were expanded to support patent creation, document upload and similarity analysis. Testing focused on ensuring accurate text extraction and embedding generation across different document formats.

Reflections week 3

Working with document processing and embeddings opened up fascinating insights into information retrieval and machine learning applications. I discovered that text extraction from PDFs is more complex than anticipated, with various encoding and formatting challenges. Implementing both online and offline embedding methods taught me about the trade-offs between accuracy and resources requirements. The offline TF-IDF approach provides acceptable results without API costs, though online embeddings from OpenAI offer superior semantic understanding. I learned about vector mathematics and cosine similarity's role in measuring document similarity. The USPTO API integration revealed the complexities of working with government data sources and handling rate limits. This week reinforced that machine learning applications require careful consideration of performance, accuracy, and cost factors. Understanding embeddings, and similarity metrics enhanced my appreciation for how AI can automate patent analysis tasks that would otherwise require extensive manual review.

Week 4

Activities week 4

Week four focused on frontend development and user interface implementation. I created the HTML pages for the landing page, login, dashboard, and case details views. CSS styling was developed to ensure responsive design across desktop and mobile devices. JavaScript was implemented for API communication, form handling, and dynamic content rendering. The authentication flow was integrated into the frontend, including session management and protected routes. I developed the case listing interface with filtering and sorting capabilities. The case details page was built to display comprehensive information including patent documents, similarity scores, and status updates. File upload functionality was implemented with drag-and-drop support and progress indicators. Navigation components were created to provide seamless movement between different sections of the application. User feedback mechanisms were added, including loading indicators and error messages. Cross-browser testing was performed to ensure compatibility.

Reflections week 4

Frontend development presented different challenges compared to backend work. Creating an intuitive user interface requires thinking from the user's perspective and anticipating their needs. I learned the importance of responsive design principles and how CSS frameworks can accelerate development. JavaScript's asynchronous nature initially caused confusion, particularly with promise handling and error management. The experience taught me valuable lessons about client-side state management and DOM manipulation. Integrating frontend with backend APIs reinforced understanding of the full request-response cycle. I realized that good UX design involves subtle details like loading states, error handling, and smooth transitions. Testing across different browsers revealed compatibility issues that required creative solutions. This week enhanced my appreciation for full-stack development and the skills required to create cohesive applications. The challenge of balancing functionality with aesthetics made me more conscious of design decisions.

Week 5

Activities week 5

This week emphasized enhancing the alert and notification system. I developed the alerts model with functions for creating, retrieving, and filtering user-specific alerts. The alert generation logic was implemented to automatically create notifications when similar patents are detected. Integration between the similarity analysis pipeline and alert system was established, ensuring users receive timely notifications. The frontend alert panel was created with real-time update capabilities and notification badges. I implemented alert status tracking to distinguish between read and unread notifications. The database schema was extended to support alert metadata including timestamps, similarity scores, and related case references. API endpoints were added for retrieving user alerts and updating alert status. Testing focused on ensuring alerts are generated correctly and delivered to appropriate users. Performance optimization was conducted to handle multiple concurrent users and alert processing. Documentation was updated to reflect the alert system architecture and workflows.

Reflections week 5

Building the notification system highlighted the complexity of real-time communication in web applications. I learned about event-driven architectures and how to design systems that respond to data changes. The challenge of ensuring alerts reach the right users taught me about user relationship mapping and access control. Implementing the frontend notification panel deepened my understanding of dynamic UI updates and user engagement patterns. I discovered that notification systems must balance between providing timely information and avoiding alert fatigue. Performance considerations became apparent when testing with large datasets and multiple simultaneous similarity analyses. This week reinforced the importance of thinking about scalability from the beginning. The patent domain aspect taught me that similarity detection has varying thresholds depending on use case and legal requirements. Understanding how patent attorneys would use such notifications influenced design decisions significantly.

Week 6

Activities week 6

Week six concentrated on integrating the USPTO data source and expanding patent retrieval capabilities. I completed the USPTO.py module in the sources directory, implementing comprehensive functions for searching patents, retrieving patent details, and normalizing data formats. The patent import functionality was developed, allowing users to fetch patents directly from USPTO by patent ID. Data normalization logic was created to transform USPTO's response format into the application's data model. I implemented error handling for API rate limits, connection failures, and malformed responses. The patent search feature was enhanced to support keyword-based queries against the USPTO database. Caching mechanisms were added to reduce redundant API calls and improve performance. Database operations were optimized to handle batch imports of patent data. API endpoints were created for triggering USPTO searches and importing results. Testing involved validating data accuracy, handling edge cases, and ensuring robust error recovery. Documentation was expanded to describe USPTO integration and data flow.

Reflections week 6

Working with external APIs like USPTO's provided valuable lessons in integration challenges and data standardization. I learned that external data sources often have inconsistent formats requiring extensive normalization. Handling API rate limits taught me about implementing retry logic, exponential backoff, and graceful degradation. The complexity of patent data structures became apparent—patents contain numerous fields with specific legal and technical meanings. I gained deeper understanding of the patent domain, including classification systems, citation networks, and legal status indicators. This week highlighted the importance of robust error handling when depending on external services. Testing with real USPTO data revealed edge cases not apparent in development with mock data. Performance optimization became critical when dealing with large patent datasets and multiple API calls. The experience reinforced that building production-ready integrations requires considering reliability, performance, and maintainability equally.

Week 7

Activities week 7

This week focused on implementing the LLM processor module for AI-powered report generation. I researched various large language model providers, comparing OpenAI GPT and Google Gemini APIs. The llm_processor.py module was developed with support for multiple LLM backends and automatic fallback mechanisms. Functions were created for generating patent comparison reports, document summaries, and similarity analysis narratives. Prompt engineering was performed to optimize LLM responses for patent domain content. Integration with the data processor was established to combine embedding-based similarity with LLM-generated insights. I implemented report caching to reduce API costs and improve response times. Error handling was added for API failures, token limit exceeded, and malformed responses. The report generation workflow was optimized to process multiple patents efficiently. API endpoints were created for triggering report generation and retrieving cached reports. Testing involved validating report quality, accuracy, and relevance to patent analysis tasks.

Reflections week 7

Integrating LLMs into the application opened exciting possibilities for automated analysis. I learned about prompt engineering techniques and how subtle wording changes dramatically affect output quality. Understanding token limits and context windows became crucial for designing effective prompts. The experience taught me about the trade-offs between different LLM providers—cost, speed, accuracy, and feature sets vary significantly. I discovered that combining traditional algorithms (embeddings) with generative AI creates more robust solutions than either alone. Working with AI APIs highlighted the importance of error handling and fallback strategies since external services can fail unpredictably. The patent domain knowledge I'd been building proved valuable for evaluating LLM output quality and relevance. This week deepened my understanding of how AI can augment human decision-making in specialized domains like patent law. I realized that responsible AI implementation requires careful consideration of accuracy, bias, and limitations.

Week 8

Activities week 8

Week eight emphasized testing, debugging, and system optimization. I conducted comprehensive end-to-end testing across all features, identifying and resolving bugs in authentication, case management, and patent processing. Performance profiling was performed to identify bottlenecks in database queries, API calls, and embedding generation. Database indexing was implemented to improve query performance on frequently accessed collections. Code refactoring was conducted to improve maintainability and reduce technical debt. I optimized the similarity analysis pipeline to handle larger patent datasets efficiently. Memory management improvements were made in document processing to prevent resource exhaustion. API response times were reduced through query optimization and strategic caching. Security testing was conducted to identify vulnerabilities in authentication, authorization, and data validation. Documentation was updated to reflect code changes and architectural improvements. User acceptance testing scenarios were developed and executed to validate functionality against requirements.

Reflections week 8

This week reinforced that testing and optimization are crucial phases often underestimated in project planning. I learned that systematic testing reveals issues that aren't apparent during feature development. Performance profiling taught me about algorithmic complexity and the importance of efficient data structures. Database optimization provided insights into query planning, indexing strategies, and denormalization trade-offs. Code refactoring highlighted how technical debt accumulates and the value of regular maintenance. I discovered that premature optimization wastes time, but neglecting performance leads to poor user experience. Security testing revealed vulnerabilities I hadn't considered, emphasizing the importance of security-first thinking. The experience taught me that building features is only part of software development - ensuring they work correctly, efficiently, and securely is equally important. This week enhanced my debugging skills and systematic problem-solving approaches. Understanding the full development lifecycle from implementation through testing to optimization provided valuable professional growth.

Week 9

Activities week 9

Week nine focused on enhancing the user interface and improving user experience. I redesigned the dashboard layouts for both attorney and client views, incorporating user feedback and best practices. The case management interface was improved with better filtering, sorting, and search capabilities. Visual indicators were added for case status, priority levels, and alert notifications. I implemented responsive design improvements to ensure optimal viewing on mobile devices and tablets. Accessibility features were added including keyboard navigation, screen reader support, and proper ARIA labels. Loading states and animations were refined to provide better feedback during asynchronous operations. Form validation was enhanced with real-time feedback and clear error messages. The patent details view was restructured to present information more intuitively. Color schemes and typography were adjusted for better readability and professional appearance. User onboarding flows were created to guide new users through the platform.

Reflections week 9

Focusing on user experience revealed how design decisions impact usability significantly. I learned that small details like loading indicators and error messages greatly affect user satisfaction. Implementing accessibility features taught me about inclusive design and legal compliance requirements. Responsive design challenges highlighted the complexity of creating consistent experiences across devices. This week reinforced that technical functionality alone doesn't create successful applications - user-centered design is equally critical. I discovered that gathering user feedback early and iterating based on it prevents costly redesigns later. Working on visual design improved my understanding of color theory, typography, and layout principles. The experience emphasized that good UX is invisible - users notice when it's missing but not when it's present. Testing with different user personas revealed assumptions I'd made about how people would interact with the system. This week enhanced my appreciation for the intersection of design, psychology, and technology in creating effective software.

Week 10

Activities week 10

This week concentrated on documentation and deployment preparation. I created comprehensive technical documentation including architecture diagrams, API specifications, and deployment guides. The README was expanded with detailed setup instructions, troubleshooting guides, and examples. User documentation was developed explaining platform features, workflows, and best practices. I implemented Swagger UI integration for interactive API documentation and testing. Code comments were improved throughout the codebase for better maintainability. Deployment scripts were created for automated environment setup and dependency installation. Configuration management was enhanced to support different deployment environments seamlessly. I prepared technical requirements documentation for infrastructure planning and cost estimation. Security hardening was performed including input validation, SQL injection prevention, and secure session management. Backup and recovery procedures were documented and tested. Final integration testing was conducted to ensure all components work together correctly.

Reflections week 10

Documentation work, often overlooked, proved essential for project sustainability and knowledge transfer. I learned that good documentation saves exponential time during maintenance and onboarding. Creating architecture diagrams forced me to think critically about system design and identify improvement opportunities. Writing deployment guides revealed assumptions about environment setup that needed explicit documentation. Implementing Swagger taught me about API-first design and the value of interactive documentation. This week emphasized that code is only part of a software project - documentation, testing, and deployment infrastructure are equally important. I discovered that explaining technical concepts clearly to different audiences (technical vs non-technical) requires different approaches. The experience of preparing for deployment highlighted considerations beyond development - infrastructure, monitoring, scaling, and maintenance. This comprehensive documentation will serve as a valuable reference throughout the platform's lifecycle and demonstrate professional software engineering practices.

Week 11

Activities week 11

Week eleven focused on advanced features and system refinement. I implemented batch processing capabilities for analyzing multiple patents simultaneously. The keyword extraction algorithm was enhanced to improve accuracy and relevance. Machine learning model integration was refined to better handle edge cases and improve performance. I developed advanced filtering options for case and patent searches, including date ranges, status filters, and similarity thresholds. The reporting system was expanded with customizable report templates and export options. Database migration scripts were created for schema updates and data transformations. I implemented audit logging to track user actions and system changes for compliance purposes. Performance monitoring was added to track system metrics and identify potential issues proactively. Integration testing was expanded to cover complex workflows and multi-user scenarios. Code quality improvements were made based on static analysis and code review findings.

Reflections week 11

Implementing advanced features demonstrated the difference between basic functionality and production-ready systems. I learned that batch processing requires careful consideration of resource management and error handling. Enhancing machine learning components taught me about model evaluation, feature engineering, and continuous improvement. The audit logging implementation highlighted compliance and governance requirements in enterprise software. Performance monitoring revealed the importance of observability in production systems. This week emphasized that building scalable systems requires thinking beyond immediate requirements to future growth. I discovered that code quality tools and practices prevent technical debt accumulation. Working on advanced filtering and reporting features showed how power users need sophisticated tools beyond basic functionality. The experience reinforced that professional software development involves continuous refinement and improvement. Understanding the full spectrum from basic features to enterprise capabilities enhanced my software engineering maturity.

Week 12

Activities week 12

The final week focused on project completion, final testing, and knowledge consolidation. I conducted comprehensive system testing across all features and use cases. Bug fixes were implemented for issues discovered during final testing. Code cleanup and refactoring were performed to ensure high code quality. Final documentation updates were completed including user guides, technical specifications, and deployment instructions. I created project presentations and demonstration materials for showcasing the platform. Performance benchmarking was conducted to establish baseline metrics for future optimization. Security audit was performed to identify and address any remaining vulnerabilities. I compiled lessons learned and best practices documentation for future reference. Final deployment preparation included creating production configuration, database initialization scripts, and monitoring setup. The project repository was organized and prepared for submission with proper licensing and attribution.

Reflections week 12

Completing the project provided an opportunity to reflect on the entire development journey. I learned that finishing a project requires attention to details often overlooked during development - documentation completeness, code organization, and deployment readiness. Final testing revealed the importance of systematic validation across all features and scenarios. This week reinforced that software projects are never truly "complete" - there are always improvements and enhancements possible. The experience of building a full-stack application from conception to deployment provided invaluable learning across all aspects of software engineering. I gained deep insights into patent infringement detection, machine learning applications, and enterprise software development. Working independently on this project developed self-discipline, problem-solving skills, and technical confidence. The challenges encountered and overcome throughout these twelve weeks significantly enhanced my capabilities as a software engineer. This project demonstrates that with proper planning, continuous learning, and persistent effort, complex technical challenges can be successfully addressed. The knowledge and skills gained will serve as a strong foundation for future professional endeavors.