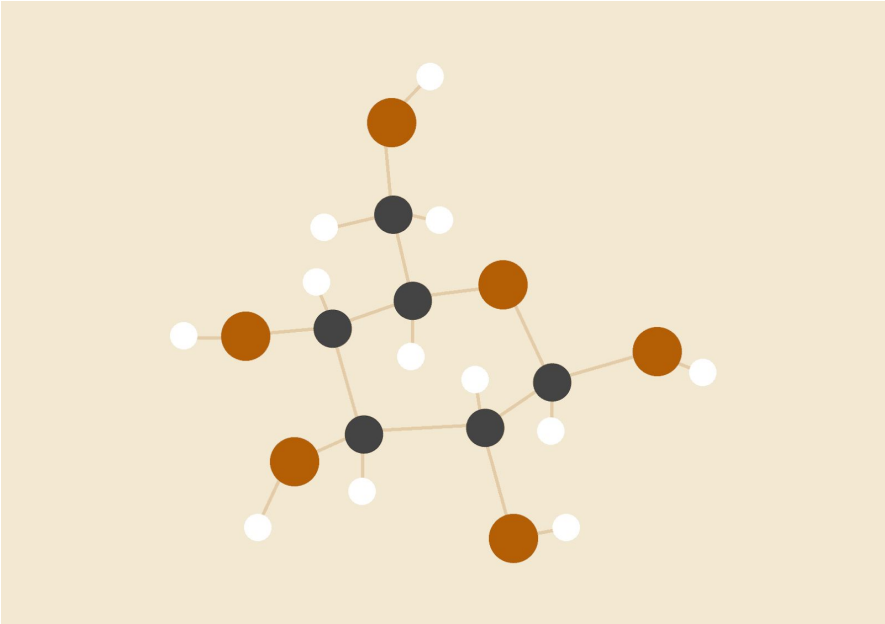
PATENTCONNECT

Software Requirements Specification

Empowering Developers, Designers, Hobbyists, & Alike



DEVELOPERS:

Yousef Hashemi <alhashey@mcmaster.ca>

Jatin Chowdhary <chowdhaj@mcmaster.ca>

Steven Gonder <gondes1@mcmaster.ca>

Yiding Li <liy443@mcmaster.ca>

Varun Verma <vermav6@mcmaster.ca>

LAB: 2 → **GROUP:** 5

The developers of PatentConnect electronically sign & date that the submitted work is strictly our own

The Domain

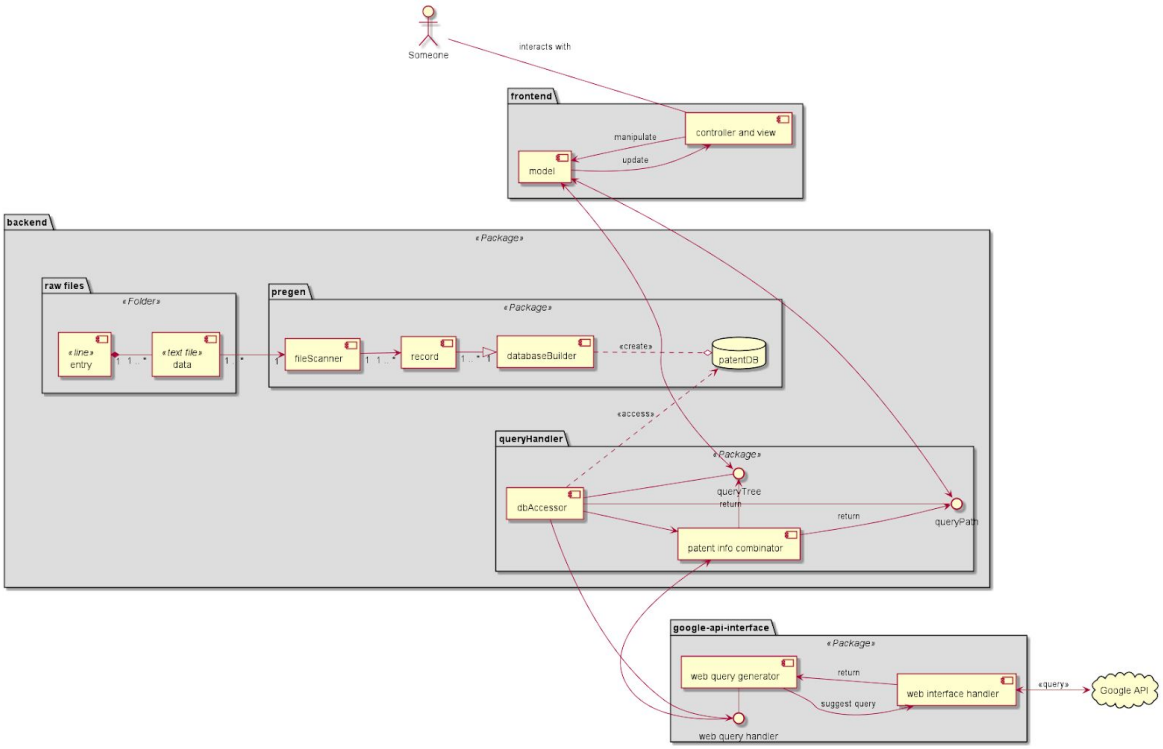
The domain. Brief description of the application domain and of the goals you should fulfill by developing an implementation. This includes a precise documentation of the domain knowledge that is relevant to derive specifications: Who are the stakeholders and what are their goals and expectations? What are the main entities that characterize the domain? What are their main relationships? How are they affected by the system we will develop?

Answer:

Functional Requirements

This product takes a patent search as the input, and provides the user with a collection of patents related to this patent, and if the user selects one of the patents they will get a description of the patent, as well as links to other patents that the selected patent references. This is achieved by the program maintaining a B-Tree that holds links to files of collections of patents. Whenever a query is made and a patent is selected the program makes a Query to Google Patents for a description of the patent to display.

Temporary UML diagram



Non-Functional Requirements

Reliability: The application should remain up unless we have a period of maintenance. These maintenance periods should not go extend further than a day. The data we pull from the APIs and datasets should not be able to be intercepted by any third parties, and we should be able to authenticate both the client and the host. We could use HTTPS secure connection and ensure that our application ensures confidentiality through encapsulation and restricted access methods.

Accuracy of results: The data should not have errors or missing data points. There is no reason for there to be a margin of error in our results, as the same result should be computed for every API call and operation performed. We will institute unit testing and integration testing along with highly specific edge case testing to ensure that the end-user will always receive the correct data.

Performance/operating constraints: Our application should have failsafe properties such that in the event of a non-termination operation (such as infinite recursion or an endless loop), we perform early termination of the operation. The operation should not crash and significantly slow the end-user's computer. The program should run on most modern household computers with ease.

Human-computer interface issues: We need to ensure that the user finds the UI intuitive. Since the application's functionality is highly specialized, we need to ensure that the user interface reflects that. Everything should be specific and unambiguous. The UI should be essential along with minimal, and have documentation for every feature.

Portability issues: Since PatentConnect will be developed using Java, we can count on the JVM's portability properties. It must work on a variety of operating systems smoothly.

Requirements On The Development & Maintenance Process

In order to create maintainable code, the development process is governed by a set of guidelines for writing, checking, and testing code. When writing code, the team will adhere to the Google Java Style Guide (Google Java Style Guide). This gives the code a professional finish and will make checking over the code much easier if all members of the team write code according to one standard. This also allows for the use of automatic style checkers such as Checkstyle.

In order to create proper code, regular code reviews will occur, where one developer reads through the code, and tries to find issues and places for improvement in the code. This will also give an opportunity for the developers to double check that the code is meeting the requirements laid out in the Non-functional requirements section. During code reviews a static code analysis tool such as CodeScan could be used to help detect bugs, dead code, or duplicate code causing bloat in the codebase.

Code testing will be accomplished through the use of the JUnit unit testing framework. Developers will write and run test code for each function they create, and the tests will try to cover edge cases, average cases, exceptions, as well as have full code coverage over each function.

It is likely that as the project progresses, certain changes will be made to the development and maintenance process. These changes are likely to occur in the tools that are being used, for example after using JUnit, and seeing that It is difficult to work with, a change in the unit testing framework may occur. Or the developers may find that the extra overhead from an automatic style checker is more distracting than helpful, and proper style is easily maintainable without it. Other than changes to the tools, the methodology behind the process is likely to stay the same throughout the project.