Introduction

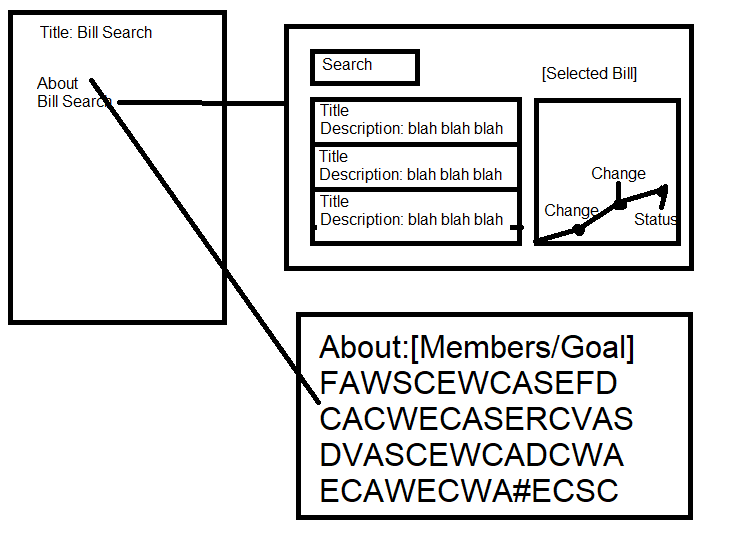
Marcus and Joseph are making a web app to display all of the Nebraska state Legislature bills. This purpose of this website is so that people can easily see and search for any bills that have gone through the state legislature.

Overall Description

The Nebraska state legislature has a website that is not very user friendly and hard to search, so we are creating a better website that will access our own automatically updating website so that people can easily get the most up to date information on the bills going through the Nebraska state legislature. Most users will simply be everyday citizens looking to be updated on what is happening in the state government. The end goal of this project is to have a fully functioning website that displays bills from the state legislature.

Specific Requirements

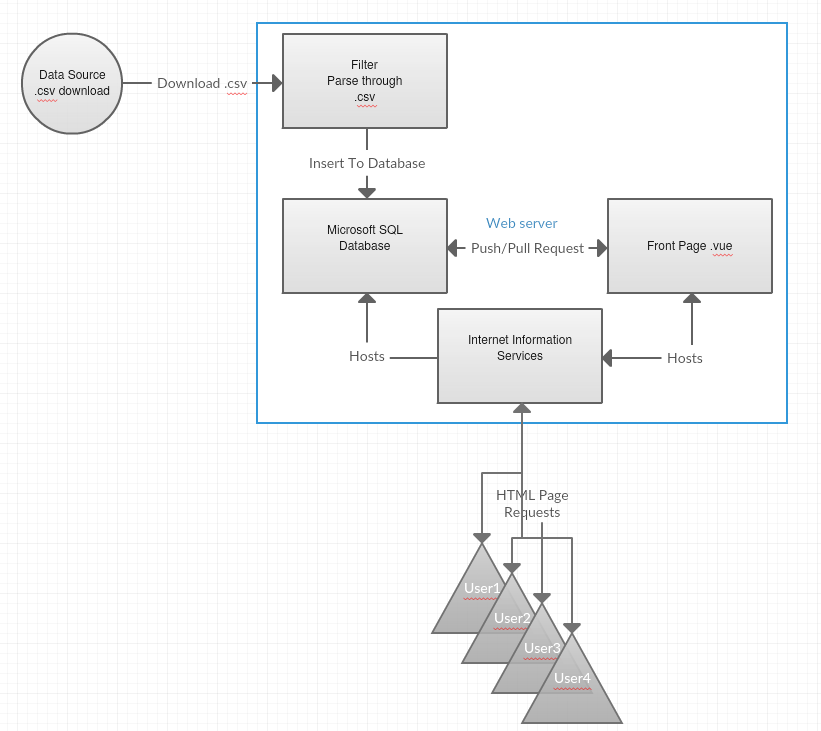
* As a concerned citizen, so that I am knowledgeable on the law, I want to search for any recent bills that changed laws on fishing over the last year.
* As a high school student, So that I get a good grade, I want to search for any bills on “gaming” for my paper on violence and videogames.
* As a layman, so I do have to learn legalese, I want search results to display an easily understandable description on what I am searching for.
* As an activist, so that I know the progress on the bill I am advocating for, I want to have a timeline of possible outcomes to know how long it might take to complete.
* As a traveling citizen, So that I can stay aware of current bills and activities, I want to be able to access and know what has changed in the Legislature anywhere and at any time.

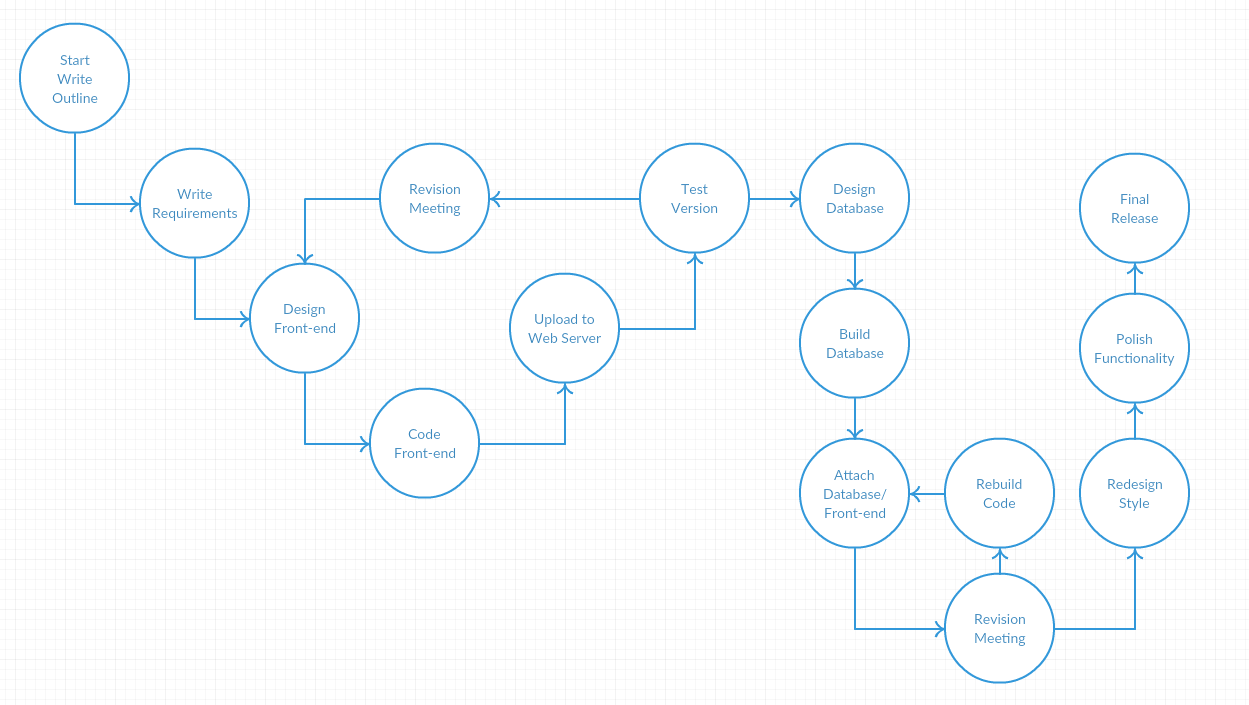
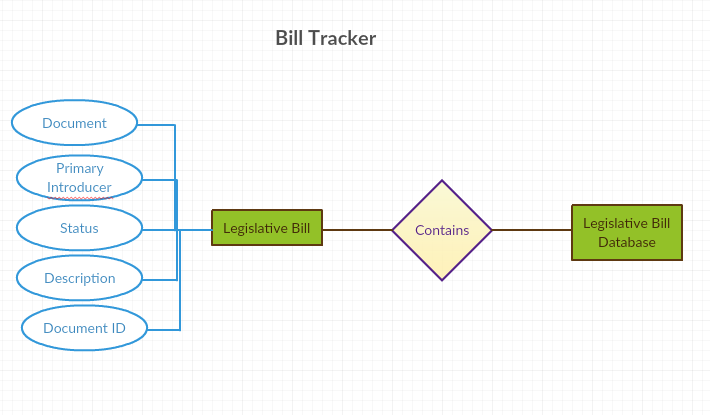
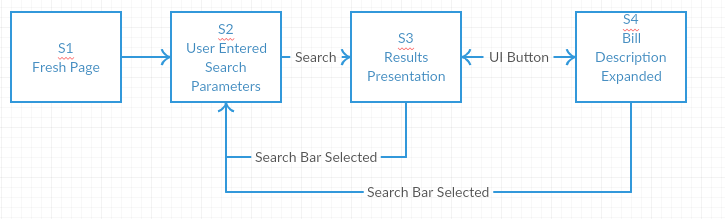
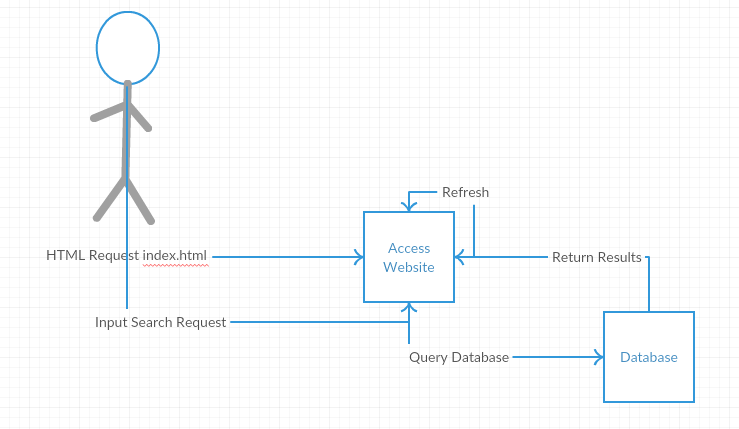


1. Introduction

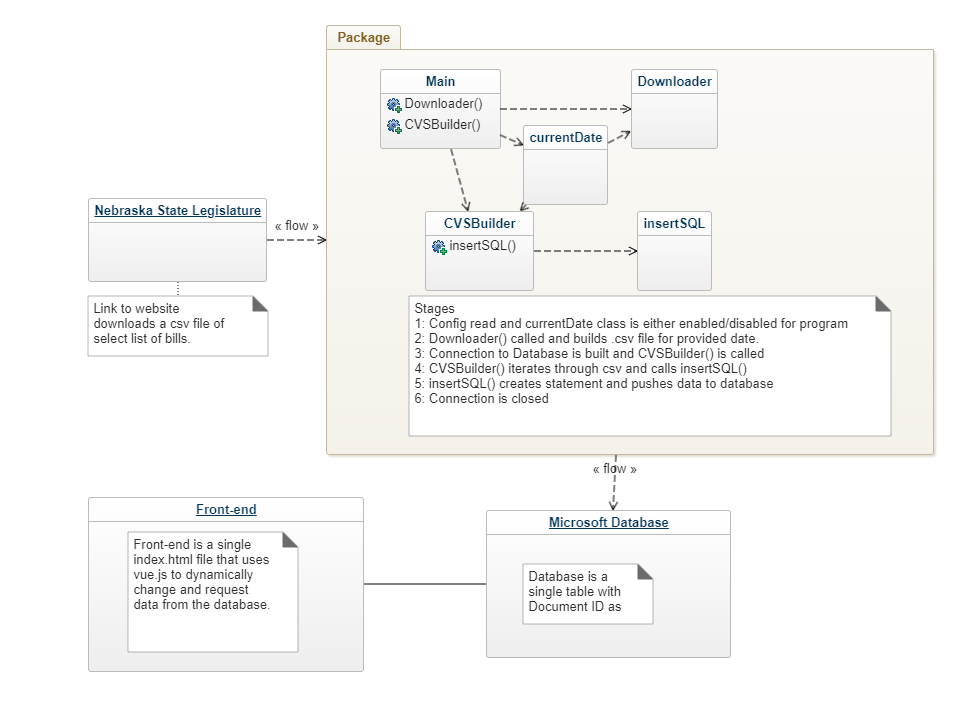
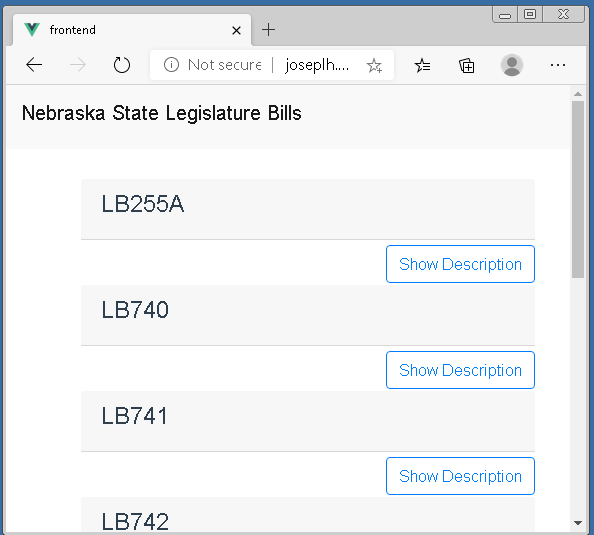
State Bills are often difficult to find and when they are found, this presented information is useless to the layman. This website and application aims to solve this problem by making searching for and reading bills easier. The Nebraska State Legislature allows users to download a quick list of all bills in a year session. By taking this .csv file we can parse through it and represent the descriptions of the bills to users.

1. Logical View of Architecture - explains the high level structure of the system's architecture; the rationale behind this structure as well as the rationale behind the choice of frameworks used. Useful diagrams include a box-and-line diagram of the logical view. This will show the logical components/subsystems to be implemented.



1. Decomposition Description (static models)
   1. Development View
      * How the development is expected to go, the front-end is designed and built first. This front-end lays out expected goals and requirements for the database and usability. After the front-end, the database is built and integrated to the front-end.
   2. Database View
      * An E-R diagram is useful here.
      * The database the application is using is very simplified. The “Legislative Bill Database” is a single entity that contains many “Legislative Bill(s)”. Each bill is composed of five attributes, Document, Primary Introducer, Status, Description, Document ID, and Date.
2. Dependency Description (dynamic models)
   * + Screen Navigation
       - Front Page only shows a search bar and drop-down menu. User can enter search parameters. Page is refreshed with list of resulting Legislative Bills. Selecting a bill will expand the description. At either results presentation or expanded bill view, if the search bar is selected the bills are minimized and refreshed based on search request.
     + Execution Flow
       - User access the database and is initially presented with a search bar/menu. User enters search request. Website queries the database. The database return results and refresh the page. User scrolls through results. User then selects a bill, expanding the description.
     + Database Dependencies: The front-end requires the ability to query a SQL database on the same server hosted. The database is built on Microsoft SQL server and the website is hosted using Internet Information Services.
     + External Dependencies – There is only one major external dependency. The Legislative website is the source of the .csv file necessary to update. Beyond the website, there are numerous minor dependencies, such as Java Development Kit, Internet Information Services, and Microsoft SQL Server.
3. Interface Description – The interface is a simple search bar and drop down menu.
4. Detailed Design – Java program used to pull and update the SQL server. Vuejs is used to build dynamic webpages that allow for multiple functions on a single page.
5. Design Evolution
   1. After implementing the web application it can be expanded to add more state legislatures. In order build this it would require additional tables for each state and separation between states within the search results.
   2. Another function would allow for a timeline for the bills to give a historical record of actions taken.

The state Bill Tracking system operates on a single windows 7 ultimate edition server housed in the basement of Joseph’s home. This server was previously used to host Joseph’s personal website which he used as a demonstration of his ability to host and maintain a live server to prospective employeers. Because of this, the front facing website is hosted on IIS (Internet Information Services), which was used previously for the personal website. The database used for the website is hosted using MSSQL (Microsoft SQL) and maintained through SSMS (SQL Server Management Studio). The website uses VUE.js to create dynamic webpages and using drivers available from Microsoft, it can pull requested data from the database. Beyond the website, on the back-end a standalone java program is used to scrape data from the state legislature website and submit it to the database. It is designed to be semi-modular and includes a config.ini file to allow for custom requests and integration.

1. Decomposition Description (static models)
2. Dependency Description (dynamic models)
   1. Process Views
      * Screen Navigation
      * Execution Flow
        + For the user, they will initially be presented a blank webpage that contains a search bar and filters. From there they can request specific information or bills. Using that data, the website then requests the data from the database and changes the page to a list of bills and description.
        + An environmental activist searches for “environment” to find what recent actions have been taken that reference the environment. The webpage filers the descriptions for “environment” and compiles a list of bills referencing the environment.
        + A concerned citizen wants to know what recently was done. Instead of entering a search term, they select latest within the filters. The webpage pulls all bills and presents the top 100 most recent based on the timestamp for the bill.
      * Database Dependencies - discusses dependencies between code modules and database tables.

There are three major dependencies that must work properly for the system to operate normally. The Nebraska State Legislature website can be redesigned at anytime and this would break the java program’s ability to pull data. The result would be that the download file would be empty and no statements would be inserted to the database. The second dependency is the connection from the java program to the database. If the connection is broken/changed, statements made from the java program would fail to upload. Finally, the front-end vue scripts also share the same dependency with the connection to the database.

1. Detailed Design - describe any nontrivial algorithms used.

The java program used has multiple modules to pull data from the Nebraska legislature and push the data to the database. The Downloader() method pulls data from the website and writes the results to a .csv file in the same folder as the java program. That .csv file is then read by the CVSBuilder() method and statements are parsed from it. The statements are sent to the insertSQL(string) method. InsertSQL then pushes the sql statement to the database. Finally the java program closes all connections and files.

1. Design Evolution
   1. Currently the design is a modular system where if we were to add more State government bills, it would be possible to redirect the java program to read from multiple sources and modify the database to organize hold multiple tables for each State Legislature.
   2. Within the java program, it reads from a config.ini file to fill out critical information needed to operate. I can insert a loop where the final lines can list all sources to download data from. There is an unused class that appends .csv files together and with the multiple sources, this class can append multiple sources together for the program to then insert into the database.