# Congressional Voting Record Database Project

**Overview**

The legislative branch of the U.S. federal government votes on hundreds of bills and measures each year and has had many different active members and parties over time. It is important for the U.S. government to keep track of their members, parties, rollcalls, and votes – it helps the public and other members of congress know what their representatives voted for to hold them accountable, and it helps the government keep attendance and track how voting patterns have changed over time.

In our project, we created a user interface to the congressional voting record database, found at <https://www.kaggle.com/voteview/congressional-voting-records>. The goal of the project is to provide software that will allow clerks in the U.S. federal legislative branch to easily track congressional voting records. The software allows clerks to record new votes, rollcalls, members, and parties and add them to a database, and also query the current database to search for past votes, rollcalls, members, and parties using certain criteria. Note that, although the database contains numerical dimensions that represent certain characteristics of a vote (how liberal or conservative a member or vote is), these require analysis by outside experts and are not used here. This project focuses on providing a useful tool for Senate and House clerks just for recording legislative branch activity and assumes that the numerical scores are updated independently.

**Functional Requirements:**

These functional requirements allow clerks to easily enter data and query data.

Add data functionality

* System must provide a navigable, readable graphical user interface
* User must be able to add data to any of the following tables of the database:
* Votes (HSall\_votes)
* Rollcalls (HSall\_rollcalls)
* Members (HSall\_members)
* Parties (HSall\_parties)
* System must require user to include all primary key values and important identifying information when adding data, to ensure quality of records
* User must be able to type in the necessary values for the record they are adding
* User must be able to query data from any table and view query in the GUI

Query database functionality

* User must be provided a simple search option to query database entries that match just one certain variable, such as members of a certain congress, or votes in a certain chamber
* User must be provided an advanced search option that provides search capabilities for specifying data items that…
* Participated or occurred in a certain date range
* Occurred in a specific congress and chamber (the two most relevant additional parameters to specify)
* Matches a member, vote, party, or rollcall that meets a certain criterion of the user’s choosing

**Schema**

This database contains four tables, or relations:

**Members**

HSall\_members(congress, chamber, icpsr, state\_icpsr, district\_code, state\_abbrev, party\_code, occupancy, last\_means, bioname, bioguide\_id, born, died, dim1, dim2, log\_likelihood, geo\_mean\_probability, number\_of\_votes, number\_of\_errors, conditional)

**Parties**

HSall\_parties(congress, chamber, party\_code, party\_name, n\_members, dim1\_median, dim2\_median, dim1mean, dim2mean)

**Rollcalls**

HSall\_rollcalls(congress, chamber, rollnumber, date, session, clerk\_rollnumber, mid\_1, mid\_2, spread\_1, spread\_2, log\_likelihood, bill\_number, vote\_result, vote\_desc, vote\_question, dtl\_desc)

**Votes**

HSall\_votes(congress, chamber, rollnumber, icpsr, cast\_code)

**Application Description and Architecture**

This application consists of a “front-end” and a “back-end.” The “back-end” handles the necessary SQL queries that must be made to the database for adding and searching data tables. The “front-end” handles the GUI display and allows the user to specify the inputs that are passed to the backend. The GUI is displayed using Java Swing. Below, each class and its functionality are listed.

*Back-end*

* AddDataEngine.java
  + Constructor: connects to the database using JDBC
  + Has methods for adding records to each table of the database
* QueryDataEngine.java
  + Constructor: connects to the database using JDBC
  + Has methods for querying data from the database for display in the GUI
  + queryTable allows for specifying a table and criteria for querying

*Front-end*

* MainMenu.java
  + Start here
  + Can open EnterDataMenu or BasicSearch
* EnterDataMenu.java
  + Starting point for entering data
  + Select a table to enter data
  + Brings you to AddNewMember, AddNewParty, AddNewRollcall, or AddNewVote
* AddNewMember.java
  + GUI for adding a new member to the database
  + Provides GUI that specifies attributes that must be included and allows users to fill in the variables
* AddNewParty.java
  + GUI for adding a new party to the database
  + Provides GUI that specifies attributes that must be included and allows users to fill in the variables
* AddNewRollcall.java
  + GUI for adding a new rollcall to the database
  + Provides GUI that specifies attributes that must be included and allows users to fill in the variables
* AddNewVote.java
  + GUI for adding a new vote to the database
  + Provides GUI that specifies attributes that must be included and allows users to fill in the variables
* BasicSearch.java
  + GUI for searching database by one variable
  + Provides interface that specifies which attributes can be searched, the data input, and the table display for the output
  + Can toggle to AdvancedSearch
* AdvancedSearch.java
  + GUI for searching database by multiple variables and a date range
  + Provides interface that specifies which attributes can be searched, the data input, and the table display for the output
  + Can toggle to BasicSearch

*Other*

* VotingRecordDB.java
  + contains the main method to start the program
* ResultSetTable.java
  + used to display a ResultSet in Java Swing
* FieldList
  + provides a data structure to pass fields to be searched to the QueryDataEngine

Below is a class diagram showing the main classes used in the application. The solid associations demonstrate how a user can navigate between the different pages in the GUI, while the dotted dependencies show how the GUI classes are dependent on the underlying engines that perform SQL queries.

Diagram

Description automatically generated

Database UML Diagram

A UML diagram of the congressional voting database (voting.db) is shown below. This diagram is also included in the “docs” folder.

*Explanation*

Each member is associated with 0..\* votes, but each vote is associated with only one member.

Each member follows one party (since if they change parties a different record for party will be created; it is considered a different member) but 0..\* members can follow one party.

For a given combination of party and rollcall, there are 0..\* associated votes. For a given vote, there are 1..\* combinations of rollcall and party (since each vote must have one rollcall and at least one party, even if the party is “unaffiliated”).

For a given combination of vote and party, there are 1..\* associated rollcalls, since each vote must be associated with a rollcall. For a given rollcall, there are 0..\* combination of parties and votes, since a rollcall can theoretically not have any votes – for instance, if the rollcall has just started and there are no votes yet.

For a given combination of rollcall and vote, there are 1..\* associated parties, since each vote must be performed by at least one party. For a given party, there are at 0..\* combinations of rollcall and vote, since a party may never actually participate in a rollcall.

Diagram

Description automatically generated