Robot Reporter Website Development

A Senior Project Report

presented to

the Faculty of California Polytechnic State University

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science in Computer Engineering

By

Samuel Verkruyse

June, 2019

Introduction

In our modern era, politics move faster than ever before. News stories come out faster and faster, and politicians are expected to be responsive at the breathtaking rate of this time. However, unfortunately, perhaps in part because of this rate, information about state level politics is in a dismal state. The practical extinction of print newspapers caused by digital newspapers has vastly reduced the number of reporters in state capitols all across the nation. Whereas there was once a proud tradition of each town's paper sending a correspondent to the capitol to cover stories that would impact their neighborhoods, there is now a dearth of coverage of these localized but nevertheless imperative stories.

Furthermore, in a time where there is more doubt in traditional news coverage than ever before, the time is ripe for an automated, fact checkable news service to emerge. Misinformation and biased articles undermine the confidence of readers. The ability of a purely unbiased bot, reliant on database inputs rather than the wills of a writer, to offer informative articles represents a unique and in demand service.

The work of Digital Democracy to retrieve as much information as possible about proceedings in various state capitols over the last few years provides the perfect opportunity to put the potential of a "robot reporter" to the test, offering information to the average layman that presently not even the most resourceful lobbyist can access.

While there is terrific potential for this project, it will require the work of a team and improvements and iterations over time. This task requires more than the work of a single individual, and continued funding of Digital Democracy must be secured to continue progress with this project. However, that is not to say that in the interim progress cannot be made, and accordingly for my senior project I sought to work on a singular facet of the project in order to continue interest in the project and demonstrate how an outsider can be brought on board onto the project and be productive working with existing code.

Throughout my project, my goal was to provide a visually appealing and functional website to interact with the output of the RobotReporter that was started by a past senior project. The website would use existing json output and a MySQL database to present the robot's stories. As I was not working with any other students, we constrained the scope of project to only UI elements, reasoning that more progress could be made if the aforementioned existing developments were utilized.

Stakeholders

The primary stakeholder for my project was the Digital Democracy project. According to their website. "Digital Democracy is a project of the Institute for Advanced Technology and Public Policy at the California Polytechnic University, San Luis Obispo". They go on to explain that the goal of the project is to expand beyond the limitations of merely providing information into the realm of enabling "individual citizens and grassroots organizations"[1]. Accordingly, I wished to make the website easy to access, share, and provide novel information not otherwise easily found.

Another stakeholder was my advisor, Professor Foaad Khosmood. As a chair of the Digital Democracy project and an advisor for past senior and masters projects for Digital Democracy, Professor Khosmood had unique insights into how past projects succeeded, could be improved upon, and what methods would have to be followed in order to ensure that this project would be useful to other future students.

Finally, the long term stakeholder of this project and any project that continues the work made by this project is proactive citizens and grassroots organizations. This was kept in mind throughout the project when deciding which features would be useful, how best to lay out information, and determining how I would make this website stand out from other news alternatives.

Increasingly, these news alternatives are relying on automated story generation. About one in every three articles by Bloomberg News are assisted by a proprietary system called Cyborg to generate stories faster than their rivals.[2] Similarly, Washington Post utilizes their own toolchain named Heliograf to free up reporting resources from easily summarized content.[3] In competing with systems such as these, Digital Democracy must offer unique, interesting and engaging content. This project will aim to do so by being a plug and play solution to create world class stories for one particular subdomain, specifically, state legislatures. The existing framework left behind from past projects distinguishes this project by making it uniquely suited for this task.

[1] See Works Cited "About the Institute."
[2] See Works Cited "The Rise of the Robot Reporter."
[3] See Works Cited "The Washington Post's Robot Reporter Has Published 850 Articles in the Past Year."

Project Goals and Objectives

Given the reasons behind the burgeoning demand for the project, the ongoing work of Digital Democracy, and the requirements of the stakeholders, there were a plethora of goals and objectives for my senior project. Given such a project with such potential, many goals were quite lofty and understandably not achievable within the duration of a single senior project completed by one student. However, they greatly informed the objectives of the project by establishing a long term vision and directing my work to lay pillars for future development.

Goals:

- Deploy a website that conveys information about local government in an interesting and automated manner
- Custom tailor the content of a news story and accompanying features according to a computer generated template
- Enable private citizens and grassroot organizations to have access to pertinent information currently only available to lobbyists and big money interests

Objectives:

- Secure a relevant domain name and a server instance in order to lay the groundwork for a long term web infrastructure
- Develop a backend in a widely known language to generate the content for the domain
- Create a visually interesting and engaging front end experience that presents the news story while highlighting important metrics and relevant external content

Project Deliverables

At the end of the project, deliverables will be put onto GitHub and made accessible to the Digital Democracy team. A particular instance of the project, reflective of the project circa the end of Spring Quarter 2019 shall be hosted on RobotReporter.me. This will be hosted on Google Cloud Engine, which will run the project until my free credits or one year passes, whichever occurs first.

Finally, brief documentation of the website shall be provided to explain the design requirements and choices, to allow future interested parties to understand the structure of the code, server, domain and to expand upon the code in whichever manner they choose.

Background

Digital Democracy was started at Cal Poly as a project of the Institute for Advanced Technology and Public Policy. This institute was envisioned by the founding director Sam Blakeslee, who after his tenure as a State Senator saw a real need for the technological prowess of Cal Poly students to collaborate with leaders in industry and politics in order to better the world through the spirit of learning by doing[4].

In particular, Digital Democracy created a never before seen infrastructure to expand clarity and accessibility of data in state governments. Originally starting in California and expanding to New York, Florida and Texas, Digital Democracy collaborated with state governments to gain access to camera feeds of state legislatures and process videos of every hearing that occurs in those states.

After Computer Science students worked to automate facial recognition and automatic transcription of voice to text, Liberal Arts students from Cal Poly worked to perfect the text output and verify the results of the algorithms. Further work was done to store these results in a MySQL database. This database contains more than 20GB of information stored across 173 tables.

A prior senior project done by Paula Ledgerwood worked to leverage the information prepared by Digital Democracy stored within this database in order to initialize the RobotReporter project. The script would take in the name of a bill and then test several templates for news stories to determine which best matched the content of the bill.

The output of this script was stored in two manners, one a set of json files, and another a simple, local HTML output. While the content was at times rather good, it became a concern that no matter how good the output became, nobody would be interested in using it if it was visually unappealing, and furthermore that it was certainly not going to be used if it was not available on the internet.

Given that my time working on the project is relatively short, this seemed a particularly ripe area of development for a senior project. My experience in my Capstone Project with web development gave me the experience needed to deploy a web server and write the code to properly display the content of the json files. Furthermore, developing immediately in the aftermath of my Capstone Project left me with an opinionated view on what aspects of web development be done better and which aspects of web development to retain during the development of this website.

[4] See Works Cited "About the Institute."

Formal Project Definition

In coming up with a formal project definition, the main idea under consideration was what aspects of current news websites that I use are enjoyable, and which areas could be improved upon. Accordingly, the list below delineates a number of fairly common sense objectives for the project, with clearly areas to stretch and improve if the customer requirements were met early. Additionally, certain requirements were included to ensure that the project could continue to be relevant after I graduate.

Customer Requirements
The website shall display all stories placed within a designated folder
The website shall be accessible from the internet from a registered domain
The website shall be useable on a variety of computing devices
The story specific pages shall be available without modification for California stories
The modifications needed to expand the project beyond California shall be noted
Web pages shall be quick to load and render
Web pages will be available to share easily
Video of the bill content shall be included if available
Dynamically generated graphs shall be included on the story pages
Links to applicable content hosted on other domains shall be included

Engineering Requirements

Along with a formal project definition, specific engineering requirements were made. As a website is not a physical entity, the delineation of engineering requirements is somewhat different as compared to an electronic product. Risk is much more relative, at least unless this project happens to garner widespread attention. However tolerance from users may be much lower as internet users have certain expectations of reliability and performance. Compliance also is much more reliant on inspection and analysis as I only have my sample size of json documents to work with, and it is through looking at this sample I will test these requirements and hope to ensure future files are compatible.

Engineering Requirements					
Spec Number	Parameter Description	Target	Tolerance	Risk	Compliance
1	Front page stories	1 per 2 json files	+/- 1 per 10 json files	L	A,I
2	Accessible domain	1 404 per 100 clicks	Min	M	A,T,I
3	Responsive design	1 rendering error per 10 devices	Max	L	I
4	California pages functional	0 404s per 100 clicks	Min	M	I
5	Notes for modifications	1 comment per change required	Min	H	I
6	Page loading time	5 seconds	+1 second	H	A
7	Share functionality	1 share section per story page	Min	L	S
8	Video embed	1 video per page where available	Min	H	I
9	Dynamically generated graphs	2 graphs per story page	+/- 1 graph	M	A,I
10	Links to applicable content	2 links per story page	+/- 1 link	M	A,I

Design

As much as any physical project, if not more so, web development relies on an intensive design process. Many design choices must be made early on, and design decisions in one aspect may very well propagate throughout the other aspects. My experience with my Capstone project taught me the necessity of using intuitive technologies, and to take advantage of my liberties when a choice had not already been pressed upon me.

When it came to Web Frameworks, I only had prior experience with Vapor, a relatively new and obscure framework created to take advantage of the new compatibility of Swift with Unix. Although my experience with Swift in iOS Development eased the transition, I grew to dislike the lack of documentation and the verboseness of the code. Having all but written off Vapor, the most compelling alternatives were Django and Express.js. While Express.js offered some exciting features, I opted instead for Django due to familiarity with the Python language and its reputation for ease of use.

Web Framework Comparison		
Framework	**Advantages**	**Disadvantages**
Django	● Well documented ● Lots of resources on StackOverflow and various other tutorials ● Python is well known by Cal Poly students	● Does not always scale well, ● Past group had difficulty integrating Javascript frameworks ● No past experience with Django
Vapor	● Prior experience during Capstone Project ● Strong typing system ● Strong performance metrics	● Steep learning curve, little documentation ● Very difficult to find any references on StackOverflow ● Few students know Swift
Express.js	● Same language across frontend and backend ● Well documented on StackOverflow and other tutorials	● No experience with Javascript beyond APIs ● Steep learning curve

As with web frameworks, I had only had prior experience with one of the options. While I have utilized Bootstrap for CSC 307, my Capstone Project and my own personal website, I had some minor gripes with its sometimes confusing hierarchy system and inconsistent naming and modifier system. While I briefly entertained doing all front end designs by myself, I found it difficult to manage all of the CSS by myself while also having enough time to generate compelling content. I subsequently took a chance with Bulma, which though not flawless gave me the opportunity to expand my repertoire.

UI Framework Comparison		
Framework	**Advantages**	**Disadvantages**
Bootstrap	● Well documented ● Prior experience using on Capstone Project ● Built in Javascript functionality	● Relative ubiquity makes pages look more generic ● Requires jquery by default, reducing Javascript customization
Bulma	● Strong library of elements to incorporate into pages ● Good documentation on website ● Easy to learn	● Less third party documentation ● Certain elements such as carousels are unavailable
None	● Unparalleled in terms of customization ● Not dependent on any third party code	● Requires lots of knowledge of CSS and HTML ● Fewer tools to make elements responsive for mobile ● Requires more testing for alternative browsers

File storage required the least consideration of any of my design choices. While technically speaking Django comes with a built in SQLite instance, and I could have converted the required tables for the project to the correct format, this could lead to future compatibility issues and double the work any time a future developer would want to add a new table for use to the server. Despite this, it was also clear that without the usage of local files it would be extraordinarily difficult to generate the pages for the website. While the original RobotReporter script could have been integrated into the Django instance and directly interact with the views without having to store the json anywhere, this was avoided as to decouple the front end from a particular version of the json generator and to avoid slow load times. This way, any json, generated in any manner, can be utilized with the project as long as the prerequisite fields are provided.

File Storage Comparison		
Storage	**Advantages**	**Disadvantages**
MySQL	● Large database of information ● Easy to query from any framework ● Secure	● Does not include any of the RobotReporter json, requires script ● Entire database requires a large amount of disk space ● Called every time page is called
Local Files	● Less overhead ● Updates very quickly, requires less knowledge than SQL ● Can be generated quickly	● Cannot access information in database beyond that contained in the local files ● Must have at least one file for every story shown
Both	● Breadth of information from the database with specificity of the local files ● Allows for cross validation and utilizing file features for lookups	● Database must be called when creating local files and calling page ● Most storage space required

Hosting ultimately became a matter of personal preference. For a relatively small scale project, all of the major web hosts are more than adequate, and GCP was ultimately chosen for exposure to a new environment and the incoming Google grant.

Hosting Comparison		
Host	**Advantages**	**Disadvantages**
AWS	● Experience using for senior project ● Most widely used web host ● Lots of tutorials and documentation	● More restrictive free tier ● Fewer discounts for students
GCP	● Easy to get free credits as a student ● Integrates well when Google grant comes in Fall 2019	● Less documentation than AWS
Azure	● Offers similar features to AWS and GCP	● Most restrictive free tier ● Least amateur and student focused of the three major hosting companies

The choice of Domain Name Provider ultimately came down to cost. As Namecheap provides free domain names for students, it was an easy choice. In retrospect, I would have used Google for my domain name if I was to do this project again. The current setup requires configuration across two websites, which is potentially detrimental for future maintenance.

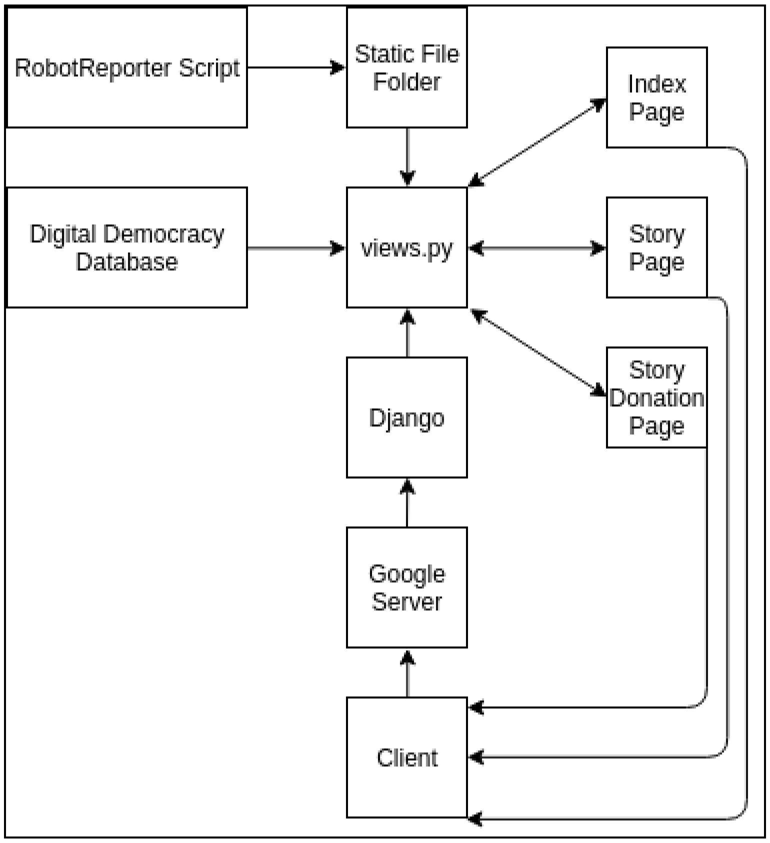
Domain Name Provider Comparison		
Domain Name Provider	**Advantages**	**Disadvantages**
GoDaddy	● Local company ● Wide range of names	● Most expensive option ● Requires a separate account
GCP	● Will transfer easier to future projects ● Available with credits ● Full range of names	● Credit can be used for hosting instead
Namecheap	● Free domain name for one year ● Easy to transfer to separate account	● Requires a separate account ● Can only get .me domain names for free

As Digital Democracy is an ongoing project, and much of its work revolves upon building on past efforts, consideration has been taken to ensure that as much of the work done by me as possible will be available to future students for utilization and avoid redundant work.

Account	**Current Solution**	**Migration Solution**
Google Cloud Platform	Running on personal account	Download backup, provide to Digital Democracy
Namecheap	Domain registered through personal account	Transfer ownership to a communal account
Server	Accessible via personal SSH key	Provide access to Professor Khosmood via another SSH key
MySQL	Accessible via root account and personal password	Utilize non root account and a new password
Passwords	Stored locally	Share all passwords and non public information via a Bitwarden account

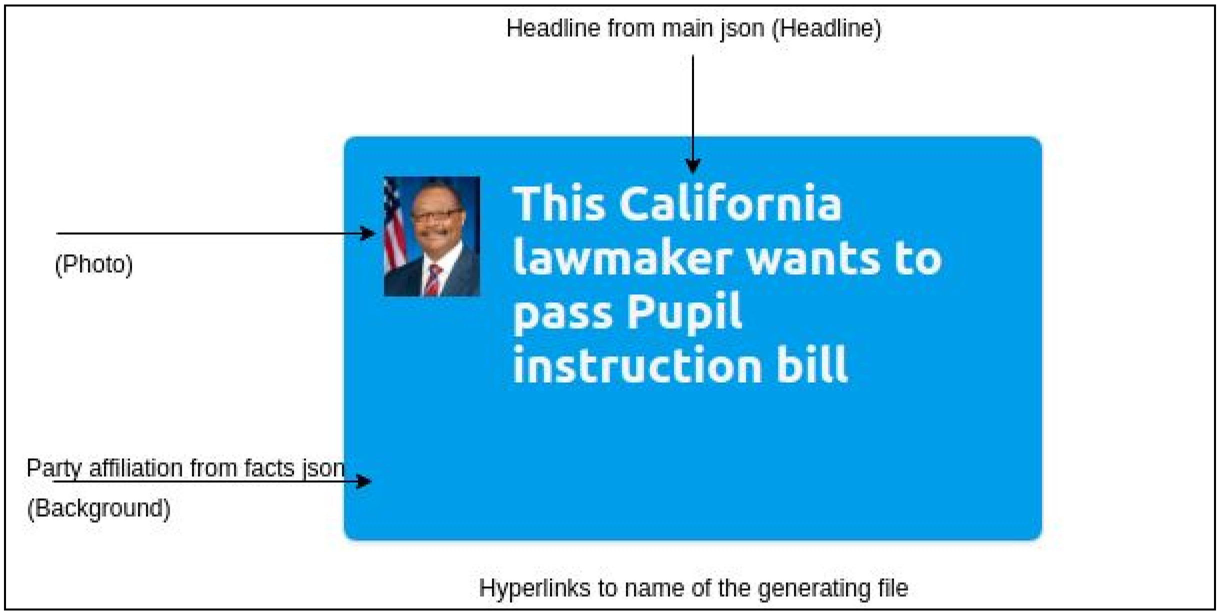
Website Breakdown

With the design decisions determined early on in the second quarter of the senior project, it finally came time to design the website. The website was envisioned to have two central pages, one index page where all stories would be displayed and the a story specific page where particulars of each story could be more closely viewed.



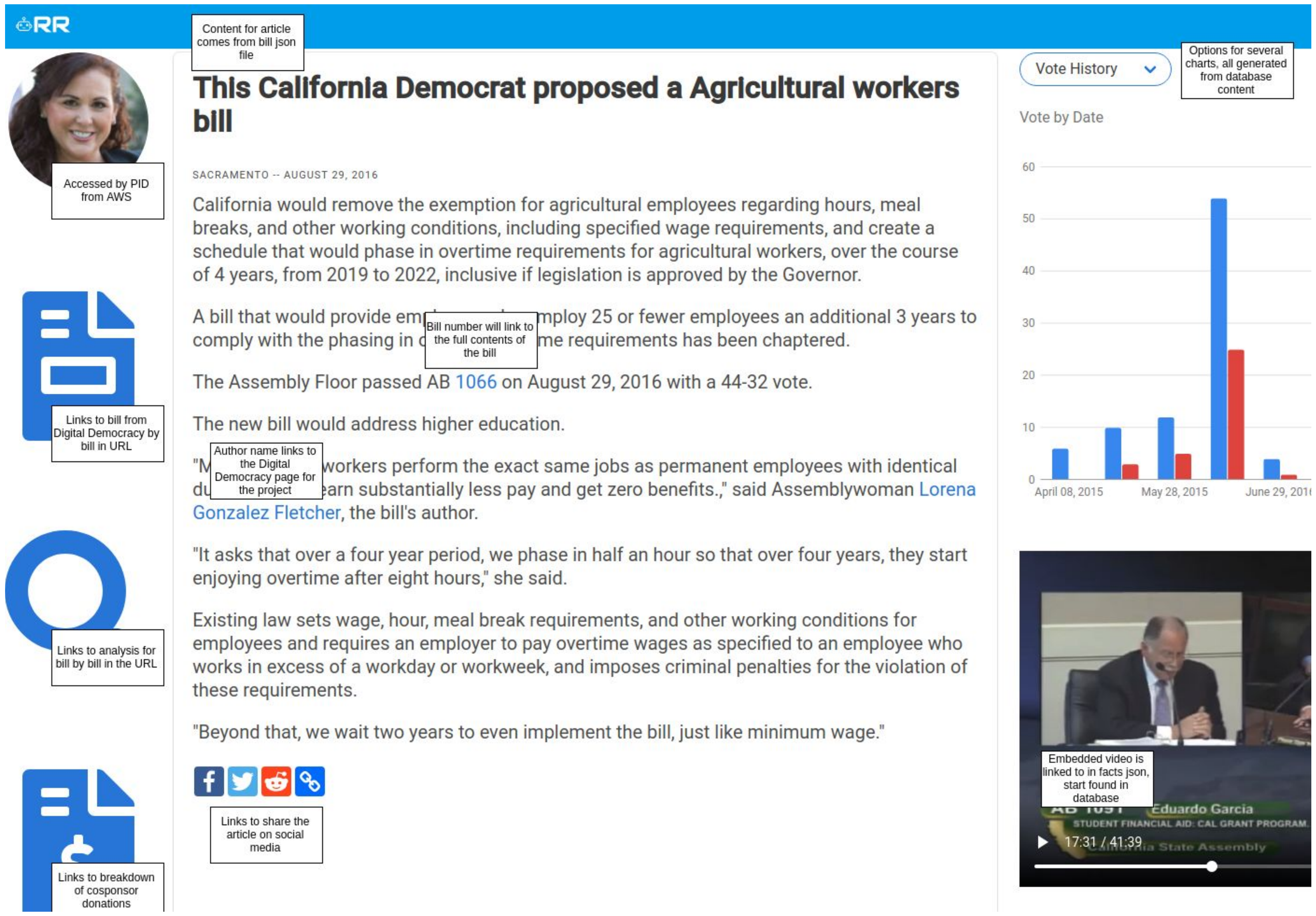
As displayed in the above diagram, the website obeys a relatively simple flow. The RobotReporter script generates the content for stories in the form of json files. While the script also creates template json and iteration json, the two relevant files for the project are the base json file and its affiliated facts file. These are placed in a special static file folder, which is provided to the views.py folder. When a client enters in RobotReporter.me, a request is sent to the Google Server, which is directed to the Django process running in a nohup configuration. As presently configured, all of the relevant URL handling code is processed by views.py. The base URL, the index page, takes in information from the files in the static file folder, and presents the viewer a grid of stories to click.

Inspiration for the front page came from cell phone app layouts and Hulu/Netflix. Rather than utilizing the conventional layout seen on conventional news websites, I aimed for a novel interface that would present news in a new and exciting way that more closely replicates modern designs rather than approximating a newspaper.


Headline from main json (Headline)
(Photo)
This California lawmaker wants to pass Pupil instruction bill
Party affiliation from facts json
(Background)
Hyperlinks to name of the generating file

Each story on the front page utilize two json files, as generated by the original RobotReporter script. One of these is the facts files. The fact files is a collection of various items and summaries of entries from the database, reflecting the information utilized to create the other json file. From this document, we utilize the pid (person identification) in order to find the picture of the representative if available, and the party affiliation of the representative in order to color code the tile. The other json file primarily contains contents for the actual story page, but the headline from this file is the text in each tile, and the name of this file ultimately becomes the name of the associated story page.

When the user clicks on one of the stories, they are redirected to a relevant story page. This story page heavily utilizes both the json files and the MySQL database. While an initial scheme utilized a precise naming scheme for these pages, a more liberal mechanism was implemented to allow for a wider array of options for titling. The base json and the facts json are loaded, and then the bill name is utilized heavily as a key to access information from the MySQL database. Namely, the actions associated with the bill are stored for a Google Chart, and the bill discussion table is accessed in order to get the start time of the video. The facts json is utilized in order to build out the vote history table, provide a link to the video. It also indirectly works to build the links to the legislator's profile, the full text of the bill, official analysis of the bill and a custom generated treemap of the bill sponsor's donations. The headline and the paragraphs of the story are accessed by the base json, and any exact reference to the representative's name or the bill's number are transformed into a link to the representative's Digital Democracy page or bill information page, respectively.



Content for article comes from bill json file

Accessed by PID from AWS

Links to bill from Digital Democracy by bill in URL

Links to analysis for bill by bill in the URL

Links to breakdown of cosponsor donations

Options for several charts, all generated from database content

Embedded video is linked to in facts json, start found in database

This California Democrat proposed a Agricultural workers bill

SACRAMENTO -- AUGUST 29, 2016

California would remove the exemption for agricultural employees regarding hours, meal breaks, and other working conditions, including specified wage requirements, and create a schedule that would phase in overtime requirements for agricultural workers, over the course of 4 years, from 2019 to 2022, inclusive if legislation is approved by the Governor.

A bill that would provide em[Bill number will link to the full contents of the bill]mploy 25 or fewer employees an additional 3 years to comply with the phasing in c me requirements has been chaptered.

The Assembly Floor passed AB 1066 on August 29, 2016 with a 44-32 vote.

The new bill would address higher education.

"M [Author name links to the Digital Democracy page for the project] workers perform the exact same jobs as permanent employees with identical du earn substantially less pay and get zero benefits.," said Assemblywoman Lorena Gonzalez Fletcher, the bill's author.

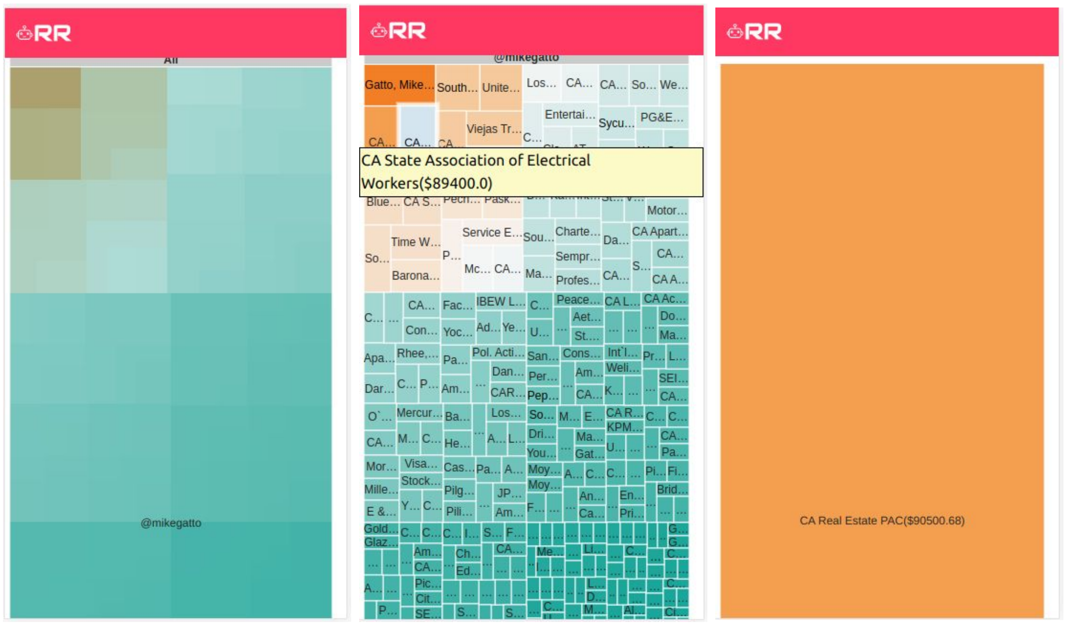
"It asks that over a four year period, we phase in half an hour so that over four years, they start enjoying overtime after eight hours," she said.

Existing law sets wage, hour, meal break requirements, and other working conditions for employees and requires an employer to pay overtime wages as specified to an employee who works in excess of a workday or workweek, and imposes criminal penalties for the violation of these requirements.

"Beyond that, we wait two years to even implement the bill, just like minimum wage."

Links to share the article on social media

The treemap page, showing donations to the sponsors of the bill, was originally part of the story page. However, due to a large amount of entries in each table and particularities with the functionality of Google Charts, it became unwieldy to have that graph on the same page as all of the other content. A design choice was made to have another page just for donations at the address of the story page followed by /donations. However, when the change came that allowed for any name to be allowed for stories rather than just the bill id, a slight modification was made so that no matter what the story was called, the donations page would be located at RobotReporter.me/{billid}/donations. This allowed for custom donations pages to be generated without any complex cookie storage, as all the information the donations pages utilizes can be accessed from the MySQL database by the bill id.


RR
All
@mikegatto
RR
@mikegatto
Gatto, Mike...
CA State Association of Electrical Workers($89400.0)
RR
CA Real Estate PAC($90500.68)

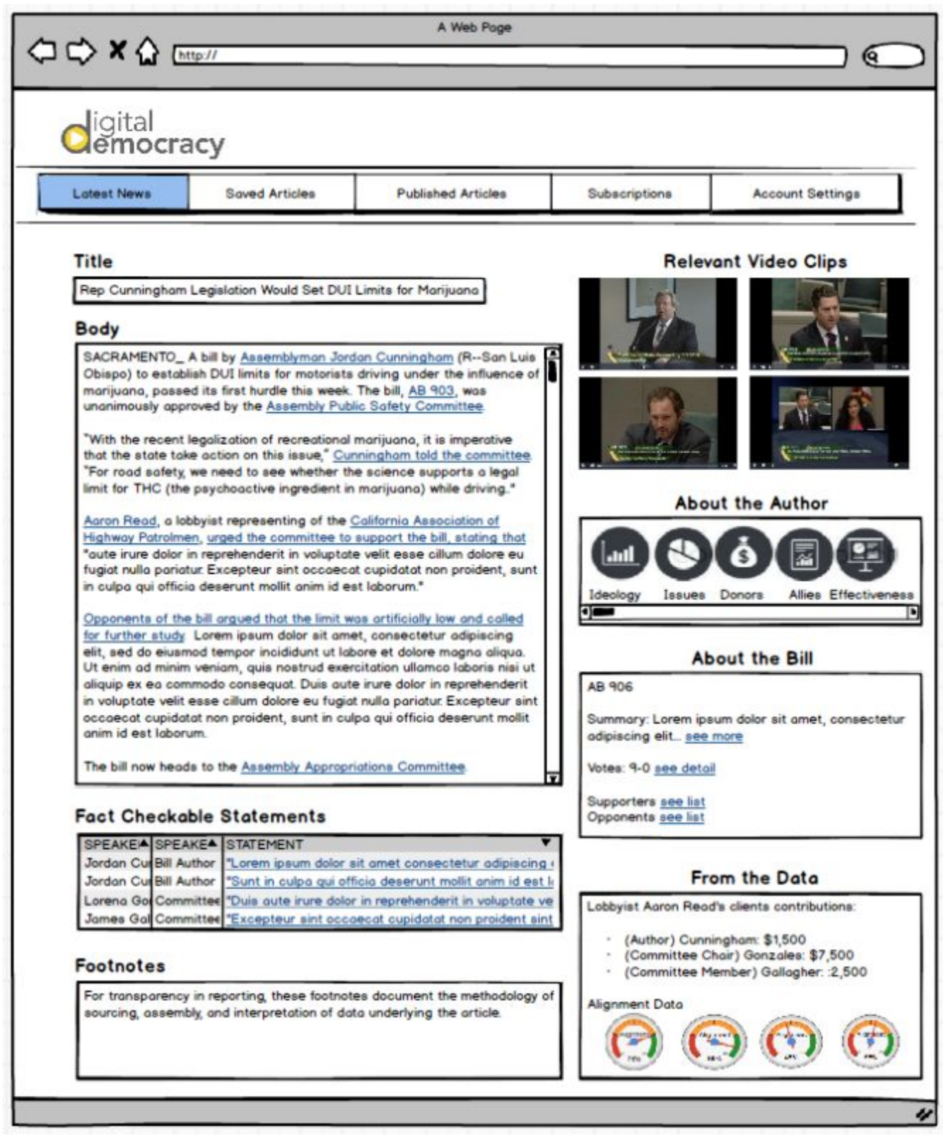
The script takes in the bill name, and utilizes the name to gather a list of all of the bill's cosponsors' pids from the BillSponsors table. A dictionary is created with the pid of the cosponsors as keys and then those keys are utilized to find the Twitter handles of each legislator, as there is no field which directly stores just the name of the legislator. Following that, the Contribution table in the database is parsed for each of the legislators, and a formatted entry is added to a string that is generated for Google Charts. After all donations are added, this string is passed to the donation page, where the treemap is displayed. The first view has an entry for each legislator, in this instance above, it is just one legislator. Clicking on the name will go into a more in depth breakdown, where hovering will show the name of the donor and the amount of the donation. Finally, clicking on any of the donations will go to a view that shows the same information as the hover caption, but all by itself.

Website Analysis

Towards the end of May, primary development on the website came to a close. With this milestone done, several hours were spent on a weekend to check compliance with engineering test requirements. Results are tabulated in a table below.

Testing Engineering Requirement					
Spec Number	Parameter Description	Target	Tolerance	Result	Passed
1	Front page stories	1 per 2 json files	+/- 1 per 10 json files	99 stories, 202 files	Yes
2	Accessible domain	1 404 per 100 clicks	Min	100 clicks, 1 404	Yes
3	Responsive design	1 rendering error per 10 devices	Max	5 devices tested, 0 rendering errors	Yes
4	California pages functional	0 404s per 100 clicks	Min	1 404 (missing twitter handle)	No
5	Notes for modifications	1 comment per change required	Min	Every setting that changes from device to device has been commented	Yes
6	Page loading time	5 seconds	+1 second	All pages render in less than 5 seconds	Yes
7	Share functionality	1 share section per story page	Min	All stories that render properly have 4 sharing buttons embedded	Yes
8	Video embed	1 video per page where available	Min	Any story that has a video url in the facts json file has a video	Yes
9	Dynamically generated graphs	2 graphs per story page	+/- 1 graph	All properly rendered pages have at least the events by data	Yes
10	Links to applicable content	2 links per story page	+/- 1 link	All stories have at least two links on the sidebar alone	Yes

Around the same time that the code came to completion, I received a forwarded email from Christine Robertson, a senior advisor for the Digital Democracy project. Enclosed was a schematic for a website highlighting certain desired features.



Title

Rep Cunningham Legislation Would Set DUI Limits for Marijuana

Body

SACRAMENTO_ A bill by Assemblyman Jordan Cunningham (R--San Luis Obispo) to establish DUI limits for motorists driving under the influence of marijuana, passed its first hurdle this week. The bill, AB 903, was unanimously approved by the Assembly Public Safety Committee.

"With the recent legalization of recreational marijuana, it is imperative that the state take action on this issue," Cunningham told the committee. "For road safety, we need to see whether the science supports a legal limit for THC (the psychoactive ingredient in marijuana) while driving."

Aaron Read, a lobbyist representing of the California Association of Highway Patrolmen, urged the committee to support the bill, stating that "aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

Opponents of the bill argued that the limit was artificially low and called for further study. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

The bill now heads to the Assembly Appropriations Committee.

Fact Checkable Statements

SPEAKE	SPEAKE	STATEMENT
Jordan Cu	Bill Author	"Lorem ipsum dolor sit amet consectetur adipiscing
Jordan Cu	Bill Author	"Sunt in culpa qui officia deserunt mollit anim id est l
Lorena Go	Committee	"Duis aute irure dolor in reprehenderit in voluptate ve
James Gal	Committee	"Excepteur sint occaecat cupidatat non proident sint

Footnotes

For transparency in reporting, these footnotes document the methodology of sourcing, assembly, and interpretation of data underlying the article.

About the Bill

AB 906

Summary: Lorem ipsum dolor sit amet, consectetur adipiscing elit... see more

Votes: 9-0 see detail

Supporters see list
Opponents see list

From the Data

Lobbyist Aaron Read's clients contributions:

- (Author) Cunningham: $1,500
- (Committee Chair) Gonzales: $7,500
- (Committee Member) Gallagher: :2,500

Alignment Data

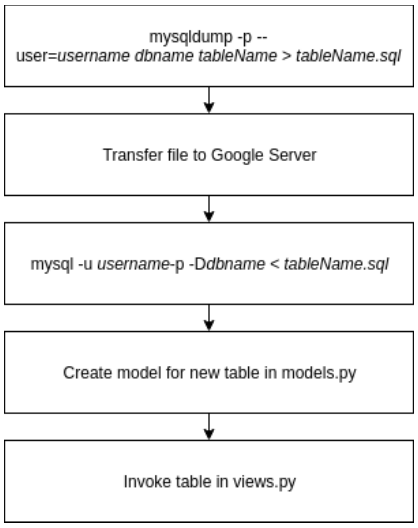
Fortunately, many of the features that Christine desired were incorporated into my design of the RobotReporter website. A few other features were added as quick functionality upgrades to the story pages. On the other hand, there were other features I did not add. The reasons these features could not be incorporated varied. Some could not be added because there was not enough information in the json to implement the feature. A few others could not be added because it would require further understanding of the MySQL schema than I had. Finally, a few features were viable, but too time prohibitive to justify development in the remaining weeks of my senior project. Below is a tabulation of features, implementation details, and explanations if they could not be included.

Feature	**Implementation or Explanation**
Article title and text	Included in the story page
Link to legislator profile page	Included when an exact match to legislator name is found and with the image of the legislator
Link to hearing video/transcript	Not included due to implementation choices, but the video for the introduction of the bill is included and timestamped when possible.
Link to lobbyist profile page	Not included due to complexity and lack of support in json, could be implemented with the lobbyist MySQL table and name matching by exact match or more advanced NLP techniques
Link to bill detail page	Included when an exact match to the bill number is found and in the sidebar of the story
Link to committee page	Not included due to complexity and lack of support in json, could be implemented with the committee MySQL table and name matching by exact match or more advanced NLP techniques
Link to organization page	Not included due to complexity and lack of support in json, could be implemented with the organization MySQL table and name matching by exact match or more advanced NLP techniques
Fact Checkable Statement	Not included due to lack of fact labelling in json
About the Author	The author's webpage and details about their donations are included
About the Bill	Various graphs about the bill are included
Social media buttons	Included beyond specification
Link to bill analysis	Included beyond specification
Events by date and vote history graphs	Included beyond specification

While it is unfortunate that not every single one of Christine's specifications could be met within the time constraints of the project, consideration has been taken to ensure that any interested party who desires to include these features shall be able to immediately work on incorporating those features rather than reinventing the wheel.

There are two major options in order to add new features in the future. The first of these is to add new fields to the json files. This could happen in two ways. The first of these is to improve and iterate upon the original RobotReporter script. The original developer of the RobotReporter script left the code in the care of Digital Democracy, and any interested party could change the script to include more information in the json files. If this was done, it would be a trivial matter to change the code in the Django instance in order to access these fields in the json file and ultimately display the fields on the website. The other way to add new fields to the json files would be to create a new RobotReporter script entirely. While the design of the website was done exclusively with json output from the original RobotReporter script, effort was made to keep it compatible with any json files that include the minimum required fields. As long as these fields are included and the json files are placed in the correct folder, the website will work to utilize them as it has no knowledge or preference for how the json is created.

The other manner in which new features can be added is by accessing more information from the MySQL database. The final version of the website utilized a minimal amount of table to speed up access and minimize storage required, but the full sized table is almost 25 gigabytes. The json files included plenty of id fields for committees, legislators, lobbyists and other applicable information that by themselves are not particularly interesting but can be used as keys to extract this crucial information by the database. Fortunately, Django integrates easily with MySQL, and adding new tables to the server database is a simple process. A dump of the table is taken from the original MySQL database, transferred and loaded into the server database, a model is made in Django and then a variety of simple interfaces are available to get information from the database.



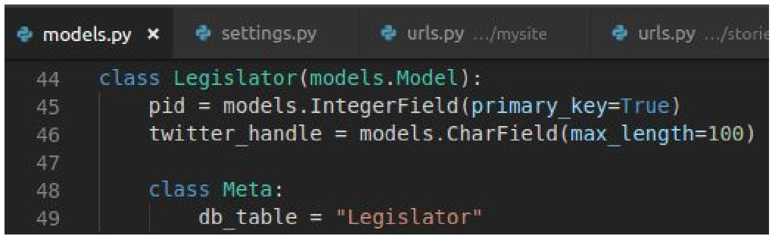
```
mysqldump -p --
user=username dbname tableName > tableName.sql
```

Transfer file to Google Server

```
mysql -u username-p -Ddbname < tableName.sql
```

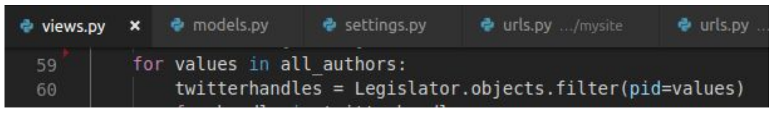
Create model for new table in models.py

Invoke table in views.py



```
class Legislator(models.Model):
    pid = models.IntegerField(primary_key=True)
    twitter_handle = models.CharField(max_length=100)

    class Meta:
        db_table = "Legislator"
```



```
for values in all_authors:
    twitterhandles = Legislator.objects.filter(pid=values)
```

Interesting Code Snippets

The entirety of the code can be found on GitHub, but per the instruction of my advisor I have included several novel pieces of code that uniquely suited some of the design challenges of developing this front end.

```
<!-- Go through each file passed and display a tile for the story -->
  {% for item in files %}
  <!-- int here determines number of columns per row. modify to change appearance -->
  {% if forloop.counter0|divisibleby:5 %}
  <div class="tile is-ancestor">
    {% endif %}
    <div class="featureRight tile is-parent">
      <!-- Modifies color of box, attempts to display photo of representative -->
      {% if item.party == "Democrat" %}
      <div onclick="location.href='./{{item.billid}}';" style="cursor: pointer;"
        class="tile is-child has-background-info box">
        {% else %}
        <div onclick="location.href='./{{item.billid}}';" style="cursor: pointer;"
          class="tile is-child has-background-danger box">
          {% endif %}
          <div class="media">
            <div class="media-left" src="{{item.photo}}" onerror="this.style.display='none'">
              <figure class="image is-48x48">
                <img src="{{item.photo}}" onerror="this.style.display='none'">
              </figure>
            </div>
            <!-- Displays the headline passed in by views.py -->
            <div class="media-content">
              <p class="title is-4 has-text-white-ter">{{item.headline}}</p>
            </div>
          </div>
        </div>
      </div>
    {% if forloop.counter|divisibleby:5 %}
    </div>
    {% endif %}
    {% endfor %}
```

This snippet is utilized in order to tile the stories on the front page. The trick behind this code comes from the Django templating tools, which include among other variables two counters, namely forloop.counter and forloop.counter0. This is incremented each time the item in files code is run. Bulma, the HTML framework I utilized, requires a tile is-ancestor div for each row. However, this div should not be closed until the desired number of tiles are inserted, otherwise it ends prematurely. By initializing the div with the forloop.counter0 divisible by 5 condition, it is immediately implemented with the 0, whereas the forloop.counter starts at 1 and will not be resolved until 5 story tiles have been inserted. After this condition is met, forloop.counter0 will be at 5 and will start a new row.

```
   #Go through the donations of the legislator and match them to the format required by Google
Charts
    chart = "[[\'All\',null,null],"
    authordict = {}
    for values in all_authors:
        authordict[values] = None
    for values in all_authors:
        twitterhandles = Legislator.objects.filter(pid=values)
        for handle in twitterhandles:
            authordict[handle.pid] = handle.twitter_handle
            chart += '[\'' + handle.twitter_handle + '\',\'All\',null],'

    #Add the donors to a set and add a space to the donor's name if it repeats because treemap
does not allow duplicates
    donors = set()
    for item in authordict.keys():
        contributions =
Contribution.objects.filter(pid=item).values('donorName').annotate(Sum('amount'))
        for values in contributions:
            donor_name = values['donorName'] + '($' + str(values['amount__sum']) + ')'
            while donor_name in donors:
                donor_name += ' '
            donors.add(donor_name)
            chart += '[\'' + donor_name.replace('\'','`\\') + '\',\'' + authordict[item] +
'\',' + str(values['amount__sum']) + '],'
    chart = chart[:-1]
    chart += ']'
```

This snippet displays accessing the MySQL database via Django's ORM system, and conforming output to the format expected by Google Charts. Google Charts expects json formatted data, which upon research is traditionally not passed from a Django view to a page. The solution was to create Python strings which precisely matched the form expected by Google Charts and to flag the string with a 'safe' parameter in the HTML template. The code accesses all of the legislators who coauthored the bill, and create an entry in the chart with their name, while also storing their name into a dictionary. Then the code iterates over the authors, accesses their donors, makes sure to prevent entries with the same name per Google convention, and tallies the sum donated to the legislator.

```
#Replace any mention of the author name and the bill number with a link to the legislator's
page or the bill's page
    for paragraphs in story_data['paragraphs']:
        if 'bill author' in facts_data:
            if facts_data['bill author'] in paragraphs['text']:
                if 'pid' in facts_data:
                    paragraphs['text'] = paragraphs['text'].replace(facts_data['bill
author'],'<a href=\'https://ca.digitaldemocracy.org/person/' + facts_data['pid'] + '\'>' +
facts_data['bill author'] + '</a>' )
        if 'bill name' in facts_data:
            replacer = facts_data['bill name'].split('AB')
            paragraphs['text'] = paragraphs['text'].replace(str(replacer[1]),'<a
href=\'https://ca.digitaldemocracy.org/bill/' + facts_data['bill'] + '\'>' + str(replacer[1])
+ '</a>' )
```

Towards the end of the project, the customer requested that the text of the article includes links to relevant information about the bill and the author. Due to the relative lateness of the request and the limitations of the provided json, the number of links are not particularly extensive, but are helpful in linking to the legislator's page and the full bill. Rather than passing links to the HTML template and doing complicated Javascript work in order to replace the words with hyperlink, this code looks to string match certain elements as defined in the facts json and turn them into <a> elements with links to the relevant pages. The above code provides clear examples of how to do so, and with a little work could be expanded to fuzzier matching and even more links.

Closing Thoughts

After working for two quarters on a website with a team, I was excited to see how the process would vary when working as an individual. Ultimately, I discovered that certain aspects are easier when working by yourself, whereas other parts are easier with the aid of teammates.

First and foremost, when working on a website by yourself, you are entirely on your own schedule. This means that any spare time you have is time to potentially develop. While when working in a group you're often waiting for new requests from the customer and code contributions from your colleagues, I have a great deal of creative freedom on this project and the ability to constantly churn out developments.

Furthermore, when working by yourself, design choices become a lot more personal. As discussed in the design section of this document, there were many choices that came down to preference. In the duration of my Capstone Project, there were frameworks utilized due to teammate preferences that I would have otherwise avoided myself, and when provided with code from peers I had no guarantee I would understand its functionality or that it would not break compatibility with my version of the code.

However, there are certainly portions of collaborative work that provide tremendous benefits. Code reviews ensured quality of code, while by myself a great deal of refactoring had to occur towards the end of this project to guarantee that the code was easy to follow. Additionally, teammates often served as a valuable resource when stuck on a problem. During my Capstone Project, if I did not know how to program a certain feature I could ask my teammates, or reassign the problem to them. Having different perspectives was also certainly beneficial, as I found that many times a solution I figured to be the only way was in fact just one of a myriad of methods after consulting with my groupmates.

In that vein, I believe this senior project has prepared me for my career after Cal Poly. As cliche at times learn by doing can seem, there is truly value in learning how to solve real world problems with your degree. Due to coming in with a large number of credits, my first two years at Cal Poly were relatively dry and filled with pure textbook learning. I struggled severely with the more abstract method of education, and quickly grew a sense of imposter syndrome.

Especially as a Computer Engineering major, I saw Computer Science majors in my classes who I believed to be far more proficient at programming than myself, and Electrical Engineering majors who embarrassed me with how little I felt I knew about circuits. It was not until I got to my upper level courses where practical knowledge was required that I felt that I unlocked the full potential of my Cal Poly education.

In that sense, this senior project was the ultimate culmination of my time at Cal Poly. While I deeply enjoyed my time in my Capstone Project group and I am proud of my accomplishments in that class, this senior project secured in my mind the fact that I am capable of designing a functioning software system from the ground up by myself. This is not to discount the work of past developments on this project, nor the help of my advisor Foaad Khosmood, both of which I am greatly indebted towards. Rather, with nothing but a design goal in mind I was given full autonomy, and at the end of the project I am proud to say that I have satisfactorily met the goals of my project. No textbooks were opened, no lectures were attended, no code was pulled from teammates, instead, every single line of code and every single feature in my project came from me sitting down and learning by doing. In my mind, no other feat could culminate my career at Cal Poly in such a satisfactory manner.

Works Cited

"About the Institute." *IATPP*, www.iatpp.calpoly.edu/about/.

Moses, Lucia. "The Washington Post's Robot Reporter Has Published 850 Articles in the Past Year." *Digiday*, 17 Sept. 2017, digiday.com/media/washington-posts-robot-reporter-published-500-articles-last-year/.

Peiser, Jaclyn. "The Rise of the Robot Reporter." *The New York Times*, The New York Times, 5 Feb. 2019, www.nytimes.com/2019/02/05/business/media/artificial-intelligence-journalism-robots.html.