Group 4
Kevin Blicharski
Logan Brown
Joseph Leiferman

https://gitlab.com/kblicharski/fswe-project

Red means push back

Gray indicates completed

Project Plan:

Week of March 5th: (Spring Break)

- ➔ Kevin: Create tests for functions within login and registration page
- ➔ Logan: Create tests for account recovery. Ensuring emails are being sent out.
- ➔ Joseph: Create Tests for endpoints, tests outputs given from accessing endpoint functions

Week of March 12th:

- ➔ Kevin: Create voter homepage
  - ◆ Voter should be able to see current races that are applicable to them
  - ◆ Voter should be able to select a race, and a candidate and vote.
  - ◆ Confirmation should be given with an email
- ➔ Logan: Create manager homepage
  - ◆ Should pull up races the manage
  - ◆ Should be able to check demographics of individual races
  - ◆ Tally votes from a individual race
- ➔ Joseph: Create Administrator homepage
  - ◆ Create ballots:
    - Add candidates
    - Voting time window
    - Where this ballot is available

Week of March 19th:

- ➔ Kevin:
  - ◆ Create tests for see that voter's votes are being saved correctly, and emails being sent
  - ◆ Test to make sure all races that a voter can vote on are available on the homescreen
- ➔ Logan:
  - ◆ Create test to make sure races are being pulled up that only the manager owns
- ➔ Joseph:
  - ◆ Create test to make sure ballots are created correctly

Week of March 26th:

- ➔ Kevin: Port over current react front-end to angular
- ➔ Logan: Port over current react front-end to angular
- ➔ Joseph: Implement login verifications, backend data verifications, set up tables for vote submission

Week of April 2nd:

- ➔ Kevin: Create voter homepage
  - ◆ Voter should be able to see current races that are applicable to them
  - ◆ Voter should be able to select a race, and a candidate and vote.
  - ◆ Confirmation should be given with an email

Group 4
Kevin Blicharski
Logan Brown
Joseph Leiferman

- ➔ Logan: Create manager homepage
  - ◆ Should pull up races the manage
  - ◆ Should be able to check demographics of individual races
  - ◆ Tally votes from a individual race
- ➔ Joseph: Test search UI from front-end

Week of April 9th:

- ➔ Kevin: Create a settings page
  - ◆ Allow change notifications
  - ◆ Font size
  - ◆ Page color
  - ◆ Link to Help page
  - ◆ Link to Feedback page
- ➔ Logan: Create Help page
  - ◆ Given user type create a guid to do basic functionalities on app
- ➔ Joseph: Create Feedback page
  - ◆ Allow for user to send feedback to us

Week of April 16th:

- ➔ Kevin: Create Tests for setting's functionality
- ➔ Logan: Create tests for settings' functionality
- ➔ Joseph: Create tests for Feedback page functionality

Week of April 23th:

- ➔ Kevin: Improve user experience in app (Add better UI, improve looks)
- ➔ Logan: Improve user experience in app (Add better UI, improve looks)
- ➔ Joseph: Improve user experience in app (Add better UI, improve looks)

Worksheet estimation:

Type	Description	Files Required	Fields	Number of Fields	Complexity	Function Point
Input	Voter Homepage	1. Vote functionality that casts a vote for a user 2. Homepage html file 3. Homepage css file	Which race, and candidate they want to vote for.	2	Average + Simple + Simple	4 + 3 + 3 = 10
Input	Manager Homepage	1. Homepage html file	What race they want	1	Simple + Simple +	3 + 3 +

Group 4
Kevin Blicharski
Logan Brown
Joseph Leiferman

		2. Homepage css 3. Demographic of a race functionality	to look at		Average	4 = 10
Input	Administrator Homepage	1. Homepage html file 2. Homepage css 3. Create a race functionality	Name, Area, Voting window, managers	5	Simple + Simple + Average	3 + 3 + 4 = 10
Output	Vote confirmation	1. Javascript file which takes userId, and sends email confirmation	None	0	Simple	= 3
Input	Settings page	1. Settings html 2. Settings css 3.Javascript functionality file	Web page color, Web page font size	2	Simple + Simple + Average	3 + 3 + 4 = 10
Input	Help page	1. Helpage html 2. Helpage css	None	0	Simple + Simple	3 + 3+ = 6
Input	Feedback page	1. Feedback page html 2. Feedback page css 3. Javascript file which sends feedback to our email	Email name, Feedback,	2	Simple + Simple + Average	3 + 3 + 4 = 10
Inquiry	Search/filter	1. Search html 2. Search javascript which takes inputs and sends to output 3. Javascript which takes output from db and updates page	Keywords, Filter by data (desc and asc)	3	Simple + Complex + Complex	3 + 5 + 5 = 10

Group 4
Kevin Blicharski
Logan Brown
Joseph Leiferman

None	Functionality testing	1. Create tests to each function responds correctly to input	N/A	N/A	Complex	3 * (7) = 21
RFP SUM:						90
Other Factors (Complexity and DI):						2
AFP (Other factors * RFP):						180
Productivity Factor = 3 Engineers						1.5
Overall Effort (AFP * PF) =						270 hours

Data Dictionary:

User

Data Item	**Definition**	Data type	**Notes**
id	Unique ID	Number	
userID	Unique id for every voter	Number	Generated upon creation
email	Email of voter	String	_@_._
userName	Voter user name	String	Must contain: At least 6 letters
password	Voter password	String	Must be: 6 characters in length Must contain: 1 capital letter 1 lower case letter 1 special character 6 characters long
ssn	Last five digits of ssn	String	Must be: 5 digits long

Group 4
Kevin Blicharski
Logan Brown
Joseph Leiferman

dob	Date of birth	Date	Must be in date format
role	User's role	String	Allowed to be either 'voter', 'administrator', 'manager'
Driver's license	Driver's license	String	
status	Whether a user is registered or not	String	Allowed to only be 'registered' | 'unregistered' | 'denied'
firstName	User first name	String	
lastName	User last name	String	
demographics	Json object containing user demographics	JSON	Includes: race, gender, party affiliation, age
precinctId	Unique precinct ID	number	

Vote

Data Item	**Definition**	**Type**	**Notes**
id	Unique vote ID	Number	Generated upon creation
electionId	Unique Election ID	Number	
votesCast	Includes ballot Id: candidate id for each ballot in the election	JSON	
voter	Voter ID	Number	Foreign key from voter table
time	Time of vote	date	

Election

Group 4
Kevin Blicharski
Logan Brown
Joseph Leiferman

Data Item	Definition	Type	Notes
id	Unique ID	Number	
managers	List of managers	[number]	
start	Start time	date	
end	End time	date	
locations	List of precinct Id's	[number]	Must be a list of valid precincts
type	Type of election	String	Local || State || National
description	Ballot description	String	
offices	List of Office Id's	[number]	

zipCodeDatabase

Data Item	Definition	Type	Notes
id	Primary key for accessing zipCodes	number	
zip	zip code	string	
primaryCity	Primary City of Zipcode	string	
state	State of ZipCode	string	
Latitude	Lat of zip	number	
longitude	Long of zip	number	
county	County of zip	string	

candidate

Data Item	Definition	Type	Notes
name	Name of Candidate	String	

Group 4
Kevin Blicharski
Logan Brown
Joseph Leiferman

dob	Date of Birth	Date	
party	Name of Party	String	
Vote count	# of votes for a candidate	number	
id	Unique ID	Number	

accessToken

Data Item	**Definition**	**Type**	**Notes**
userId	User in which access token is associated with	number	
created	Date created	Date	
scopes	List of strings containing scopes of accessToken	[String]	
ttyl	Determines if token is still valid	number	
id	Unique access token id	String	

Office

Data Item	**Definition**	**Type**	**Notes**
title	Title of Office	String	
description	Description of Office	String	
candidates	List of candidate ID's	[Number]	
id	Unique ID	Number	

Audit

Data Item	**Definition**	**Type**	**Notes**

Group 4
Kevin Blicharski
Logan Brown
Joseph Leiferman

action	Description of the action took	String	
time	Time the action took place	Date	
id	Unique ID	String	

CRUD functions for Administrators:

Create:
- Voter
- Manager
- Administrator

Read:
- Ballot
- Voter Information
- Manager Information
- Administrator Information
- Voter Demographic info
- Audit Trail

Update:
- Voter status
- Voter Information
- Manager Information
- Administrator Information

Delete:
- Manager
- Administrator
- Voter

CRUD functions for Managers:

Read:
- Voter Information
- Ballot Information

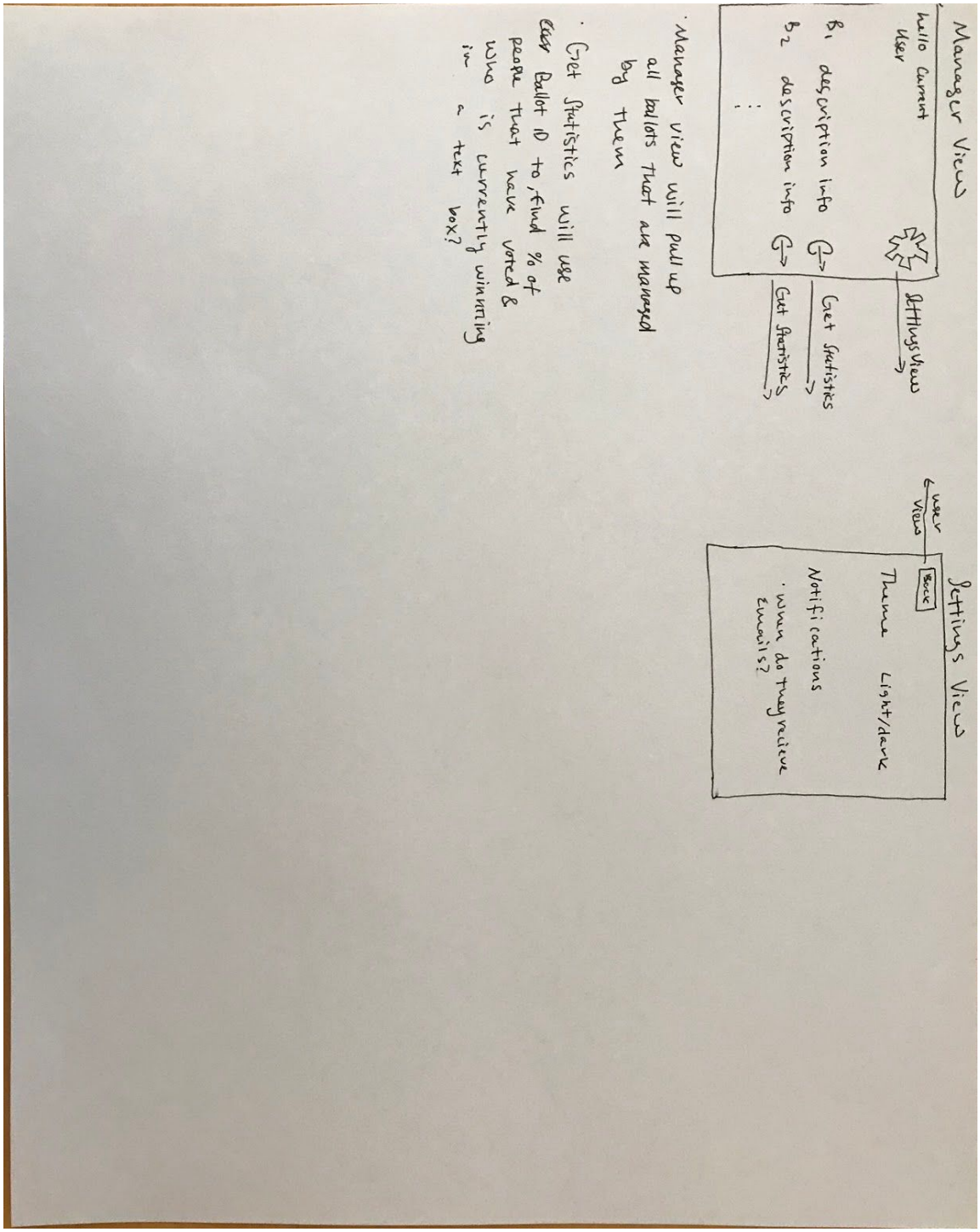
Update:
- Voter status
- Voter Information

Delete:
- Voter

Group 4
Kevin Blicharski
Logan Brown
Joseph Leiferman

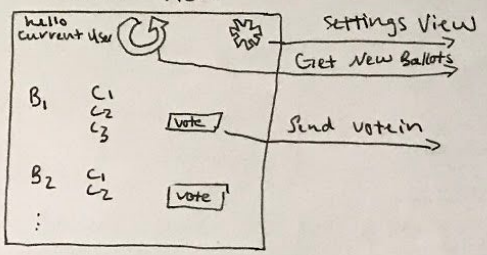
Group 4
Kevin Blicharski
Logan Brown
Joseph Leiferman

UI Screenshots



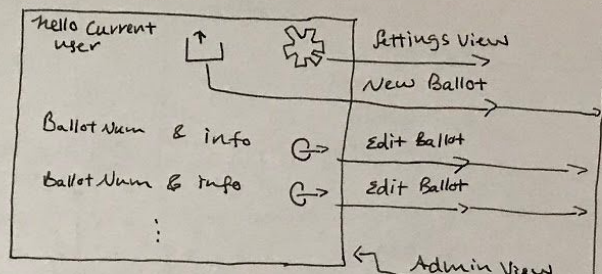
Group 4
Kevin Blicharski
Logan Brown
Joseph Leiferman

Voter View



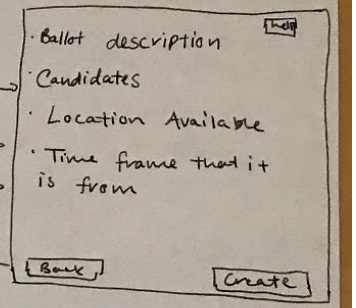
- Upon Entering Voter View front-end should query back end for ballots that correspond to the user's information
- Get new ballots takes current user info and refreshes the page
- Settings will pull setting View, where password can be changed. Notifications can be changed, or even look can be changed
- Pull up ballots that a voter can vote on, and their relative candidates. User can then click a candidate and press vote for each individual ballot.
- Vote button will send ballot ~~ID~~ Number & candidate & voterID to Post votes

Administrator View

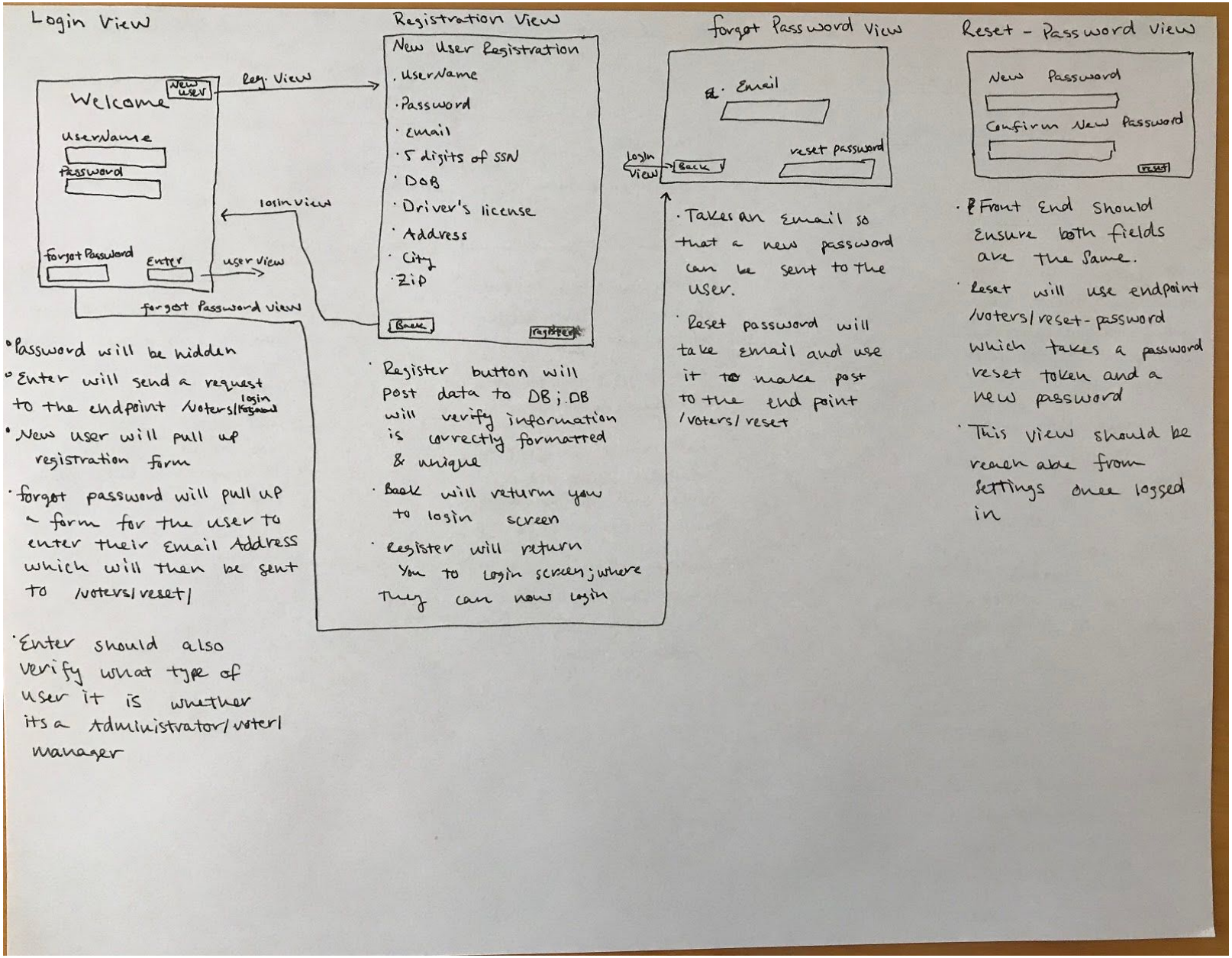


- Settings will operate the same as User settings View
- New Ballot will pull up new/edit ballot view
- Edit ballot will pull up new/edit ballot view that will be populated will current info
- Admin View will pull up ballots that were created by current Admin user

New/Edit Ballot View



- Create will use put /votes/ which will either create a new ballot and or replace an old ballot
- Help button will up text about how to enter information into each box properly.



Group 4
Kevin Blicharski
Logan Brown
Joseph Leiferman