Team 3

Zhaozhong Qi
AJ Turner
Maggie Macfadyen
Shane Clegg

Overview

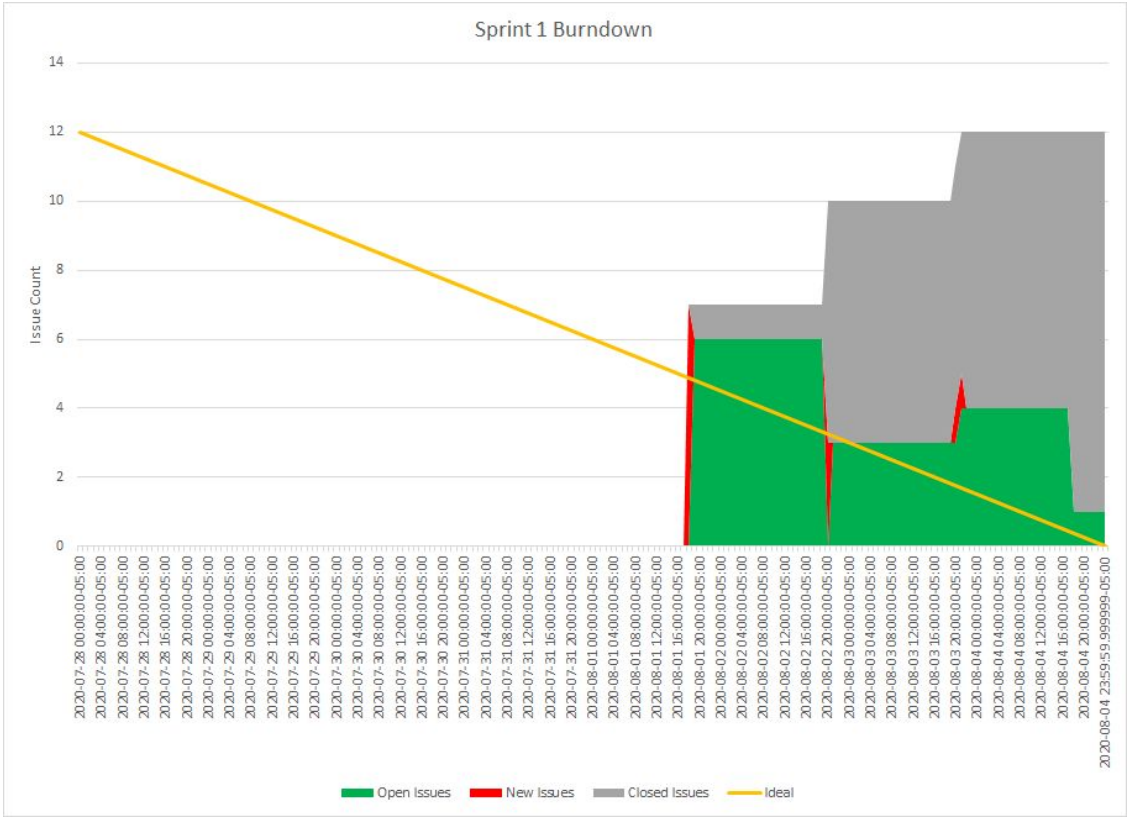
- Project Overview
- Design
- Implementation
- Demonstration
- Reflection

Project Description

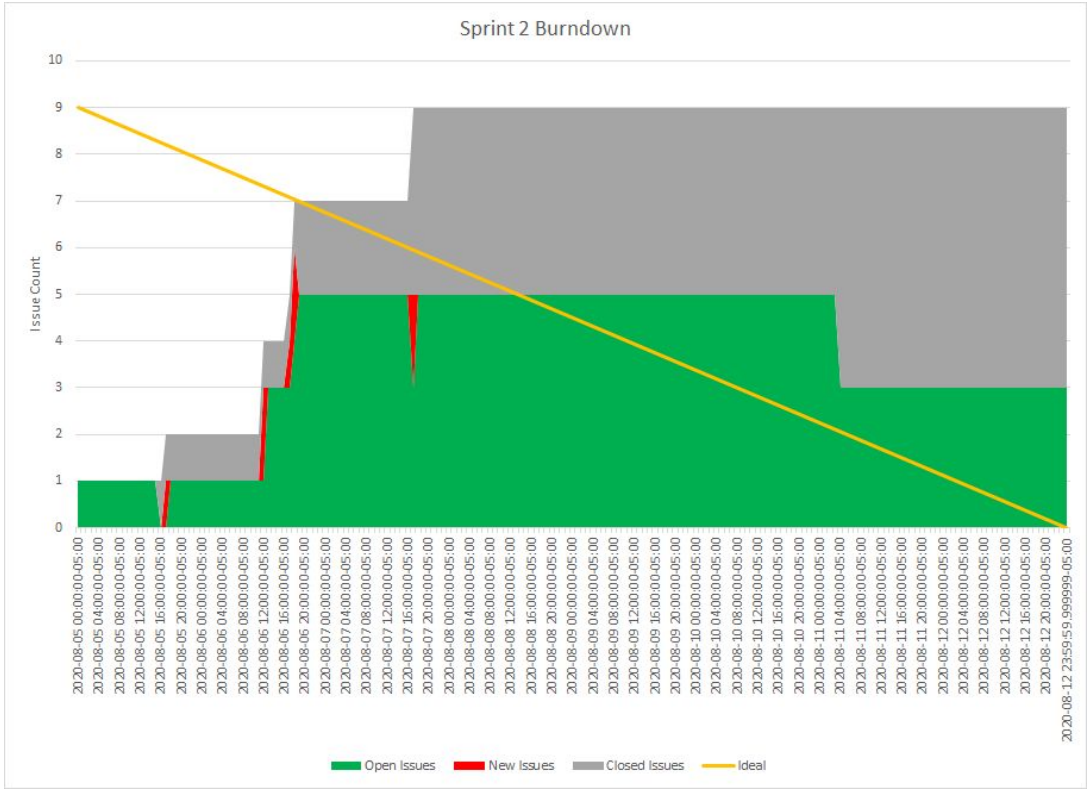
- Voting System for the city of Pacopolis
- Covers elections and issues, including races for positions and propositions
- Two types of people: Voters and Election Officials
- Voters and Election Officials can create accounts corresponding to their roles
- Election Officials can create ballots, and Voters can vote on the ballots that are created by the Election Officials
- All users can view election results at any time

Burndown Charts

Sprint 1



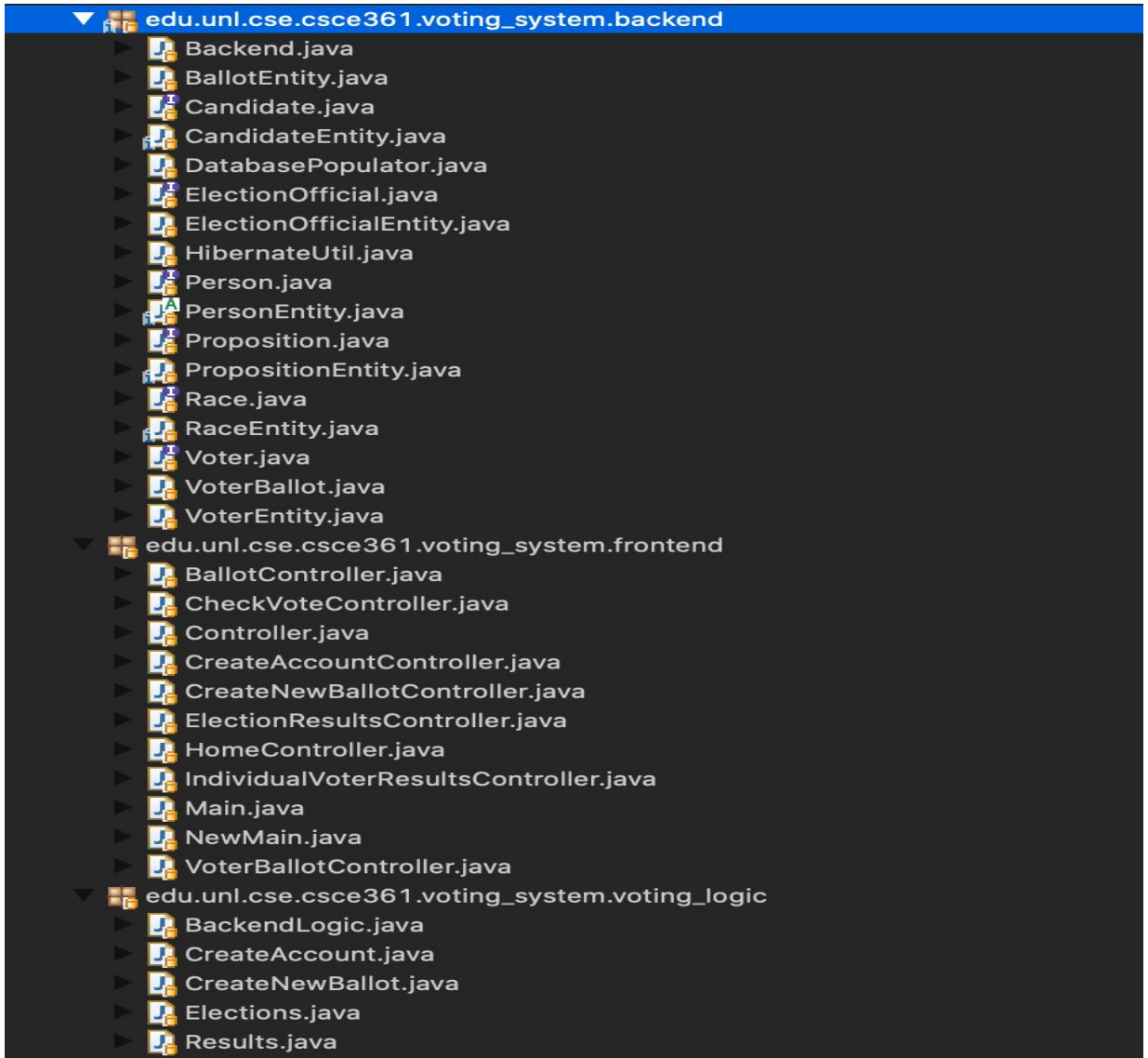
Sprint 2



Overview

- Project Overview
- Design
- Implementation
- Demonstration
- Reflection

Subsystem Decomposition

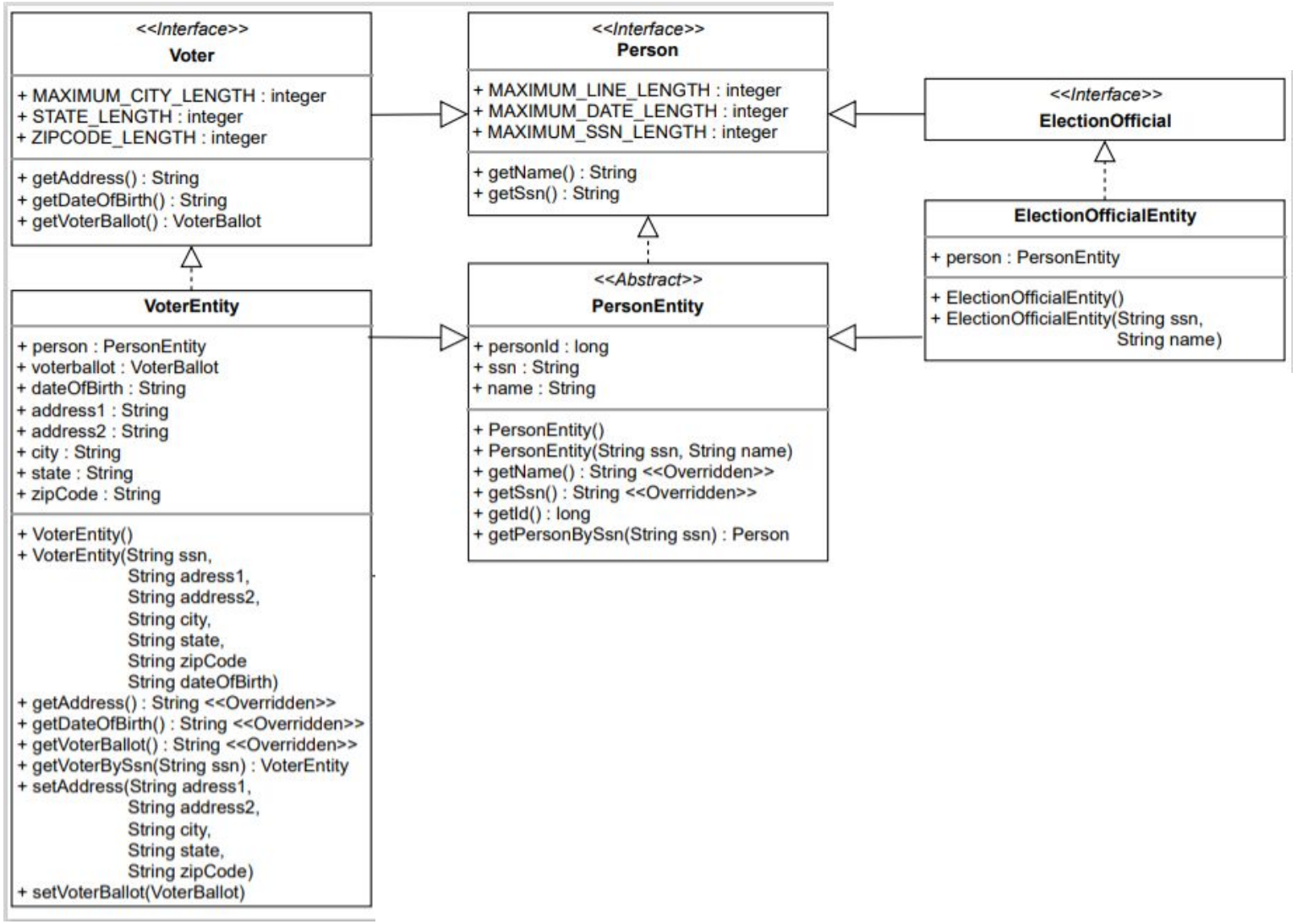


Three Tier Architecture

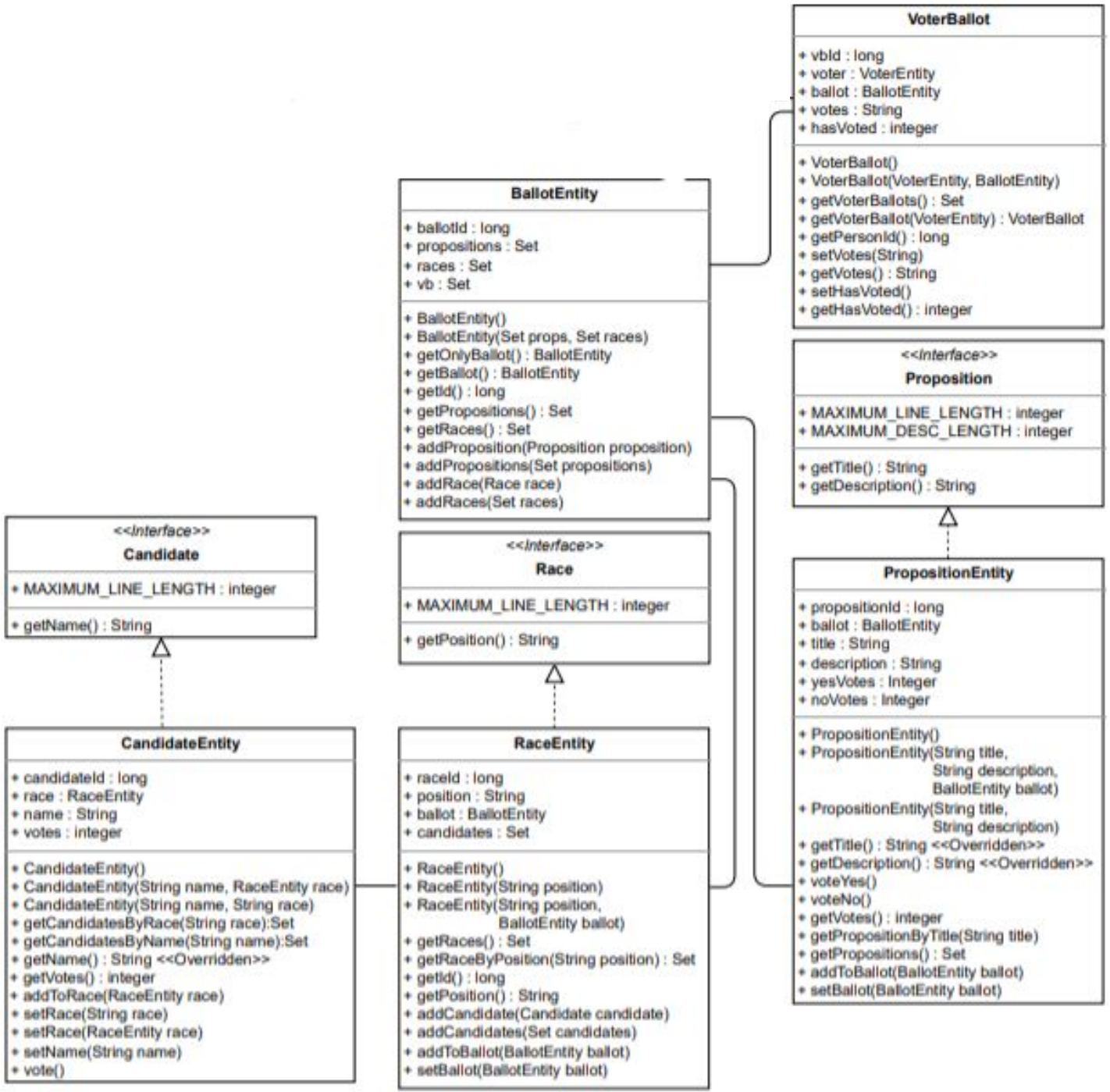
Other Architectural Issues

- Storing a voter's ballot selections in the database
- Deciding if a candidate on a ballot should be a person

Class Diagram:
Person Subsystem



Class Diagram: Ballot Subsystem



Overview

- Project Overview
- Design
- Implementation
- Demonstration
- Reflection

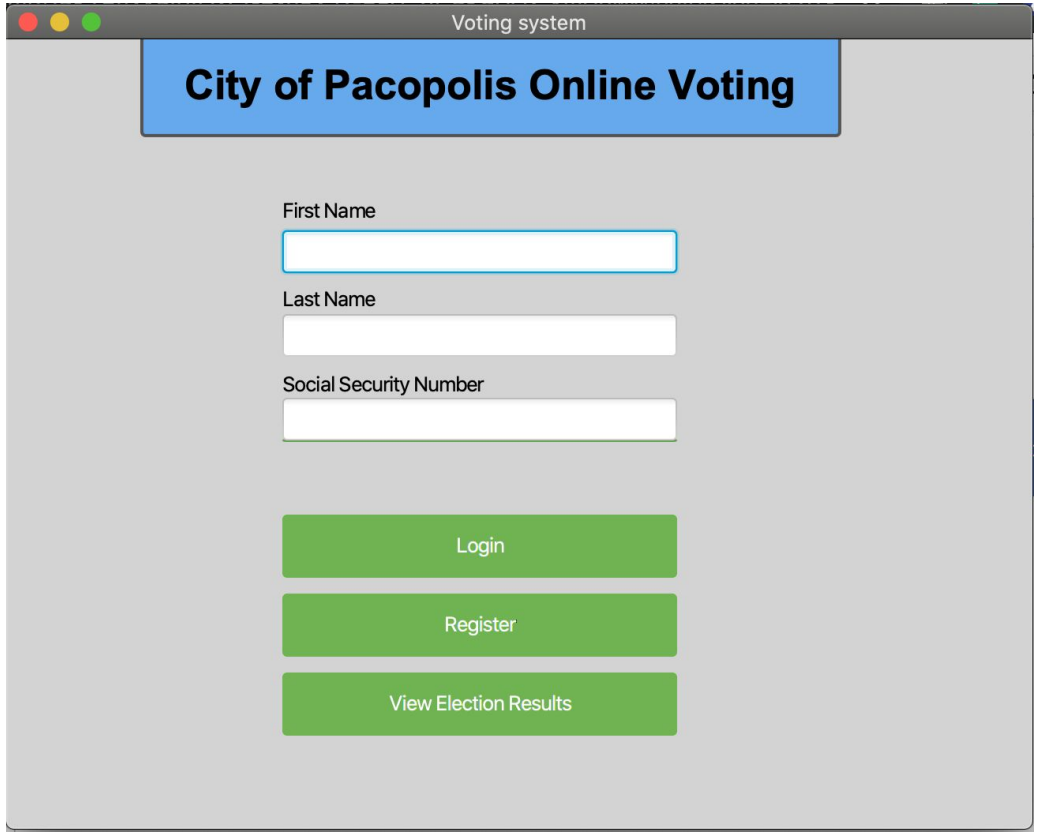
Implementation

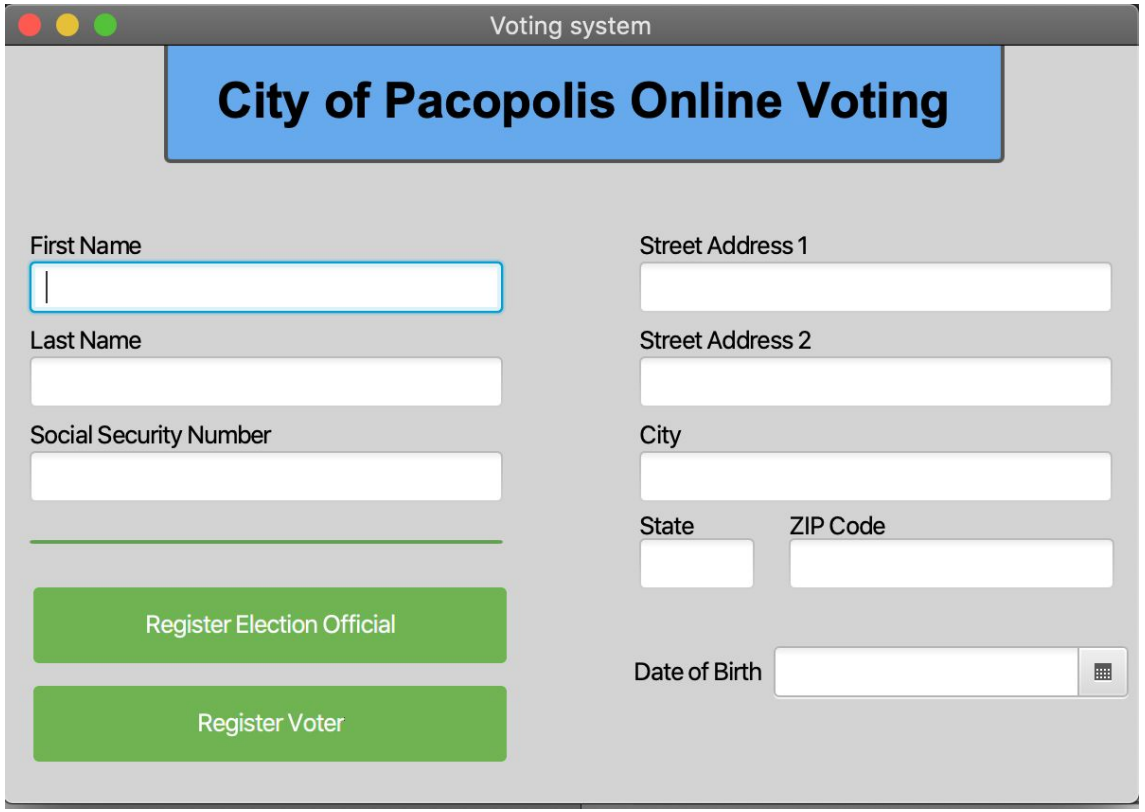
- Libraries / Dependencies:
- (other interesting info)
- Progress:

Overview

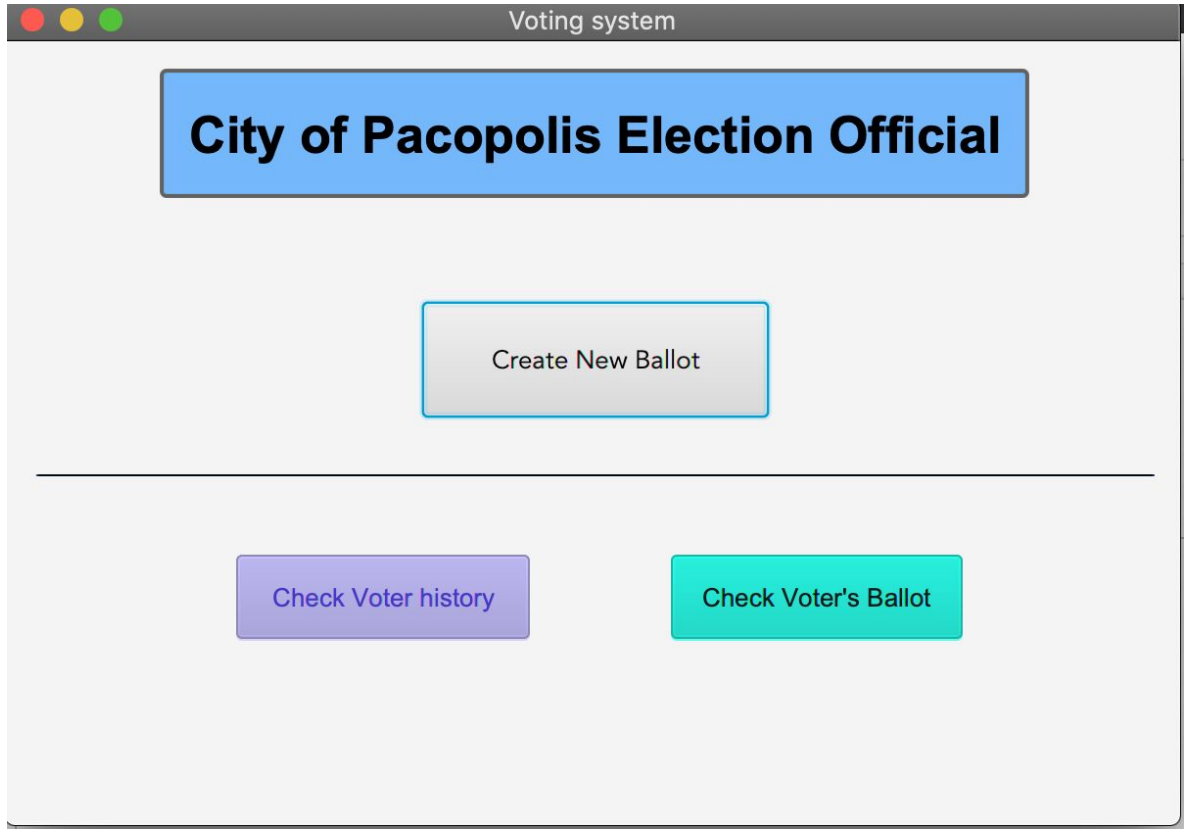
- Project Overview
- Design
- Implementation
- Demonstration
- Reflection

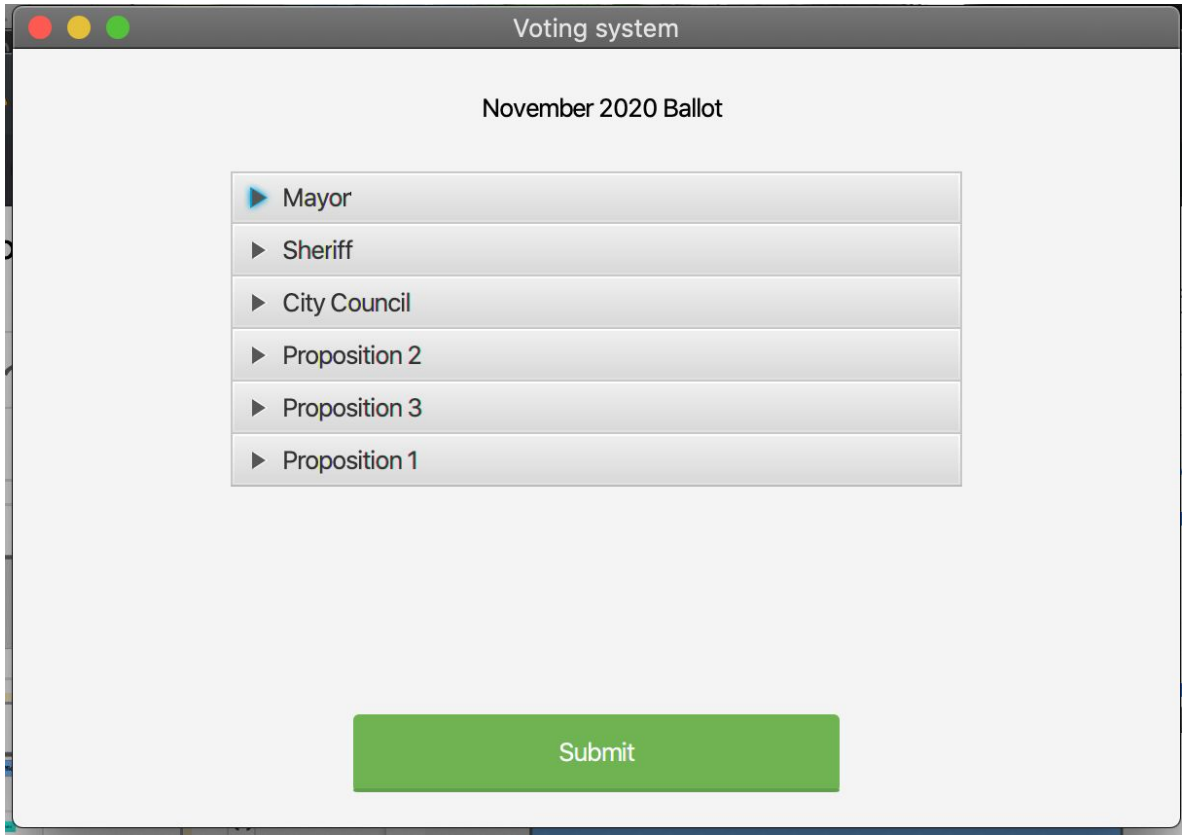
Demo


Voting system
City of Pacopolis Online Voting
First Name
Last Name
Social Security Number
Login
Register
View Election Results


Voting system
City of Pacopolis Online Voting
First Name
Last Name
Social Security Number
Register Election Official
Register Voter
Street Address 1
Street Address 2
City
State
ZIP Code
Date of Birth

Demo


Voting system
City of Pacopolis Election Official
Create New Ballot
Check Voter history
Check Voter's Ballot


Voting system
November 2020 Ballot
Mayor
Sheriff
City Council
Proposition 2
Proposition 3
Proposition 1
Submit

Overview

- Project Overview
- Design
- Implementation
- Demonstration
- Reflection

Reflection
What went well

- Using hibernate for the backend and JavaFX for the frontend went smoothly as we were familiar with both softwares
- Peer programming using screen-sharing was useful for brainstorming ideas and debugging code
- Backend objects were created first which made implementing the frontend and logic layers much easier

Reflection
What didn’t go well

- Committing code and merge requests didn’t always go well, a good deal of code was committed to branches higher than it should have been, including non-working code, which in turn made merge requests messy
- Fixing merge conflicts. In the end we had to manually compare files to fix the merge conflicts in some cases
- Connecting the Person subsystem and Ballot subsystem was difficult

Reflection
Lessons Learned

- Committing files directly to a branch that needs to be merged into is bad practice because it creates merge conflicts which can be difficult to resolve
- Tools like Git Extensions can be very useful for resolving merge conflicts when configured properly
- A clear class diagram is useful in creating one-to-many and one-to-one relationships, especially when dealing with a large number of objects