Solution Engineer Assignment

Welcome to LINE Town. Hi, you must be the new solution engineer TH-HQ sent to help us. You are in one of the meaningful events in the town. We are about to have a new mayor! Previously, we handled this event manually, but this time our population has increased significantly, and also, there are a lot of candidates, as never happened before. We know some technologies can help, so that's why you come here to help us figure it out.

After you and LINE Town discussed, You were assigned to build a web application for citizens (users) to vote for their new mayor. Coming to the first page, it will show all the candidates. Users can browse and learn about their interested candidates. And when the time comes, LINE Town's secretariat will open the vote via the new system. Then, citizens can vote for their choice, a single one and only. Everyone can see points going to each candidate during the vote in real-time. LINE Town's secretariat will close the vote following the planned schedule (on their clock) via the same system. Nobody can submit from now on. Vote's result will be summarized and replaced on the current vote page. At that moment, all citizens will know who be the new mayor of LINE town. By having the new system, this chaotic situation will be in control and make everyone in town happy.

However, this job has three different roles depending on which you were assigned at first from the TH-HQ. Each one has its own requirements. Therefore, these are the detailed requirements if you are;

A solution front-end engineer

You are more likely to focus on implementing the UI part and connecting with APIs

1. Implement a page showing the candidate list, voting, and seeing the result.
2. Citizens can use this application on their mobile.
3. Retrieve and send data through GraphQL.
4. We will allow citizens to vote right on the page when the vote is opened.
5. Voters can not vote again. We have to check their identity by citizen ID.
6. The numbers of the vote can be seen in real-time for each candidate.
7. When the vote is closed, no one can vote. The current page will calculate the score and sum up who is the new mayor.

8. The unit test is required, especially on the complex logic.

Notes:

- The wireframe is given below (Appendix #1).
- The GraphQL server is here https://github.com/pangaunn/election-api.

Bonus:

- Beautiful UI, what we give you is just a wireframe you can decorate as you the way you want, show us your sense of design.
- Introduce other new features beyond the requirement. For example, lazy loading, infinite scroll, preserved scrolling when navigating back and forth, search candidates, some charts, etc.

A solution back-end engineer

You are the guy who provides the necessary data and maintains the system's stability.

1. Implement a CRUD API for a candidate resource.
2. Implement an API for opening and closing the vote.
3. Implement an API for returning the vote result.
4. Implement API for voting. It is real-time. It cannot vote again.
5. Handle large traffic after opening the vote. You have to make your system support throughput at least ~100 RPS.
6. Implement API for exporting the vote result as a .csv file.
7. All API must be authenticated via Bearer token.
8. Unit tests are required.

Notes:

- The API Spec is given below (Appendix #2).

Bonus:

- With more throughput, more points you can get.
- Introduce other new APIs. For example, candidate search API, e.g.

A solution full-stack engineer

You are the one who mainly focuses on solving the problem without boundaries of which part you are working on.

1. Implement a page showing the candidate list, voting, and seeing the result.
2. Retrieve and send data through your implemented API.
3. We will allow citizens to vote right on the page when the vote is opened.
4. Implement a voting mechanism, and the vote can be opened or closed by API.
5. Voters can not vote again. We must check their identity by citizen ID both on the client and the API.
6. The number of votes can be seen in real-time for each candidate on the page.
7. Handle large traffic after opening the vote. You have to make your system support throughput at least ~100 RPS.
8. When the vote is closed, no one can vote. The current page will calculate the score from the server-side and sum up who is the new mayor.
9. Unit tests are required both on the front end and the back end.

Notes:

- The wireframe is given below (Appendix #1).
- The API Spec is given below (Appendix #2), and you do not need to implement all of it, just some that need.

Bonus:

- Anything, any improvements apart from the requirements above if you can manage all of it in time.

A little guidance

1. Please, read the assignment thoroughly.
2. Taking a look at the wireframe below (Appendix #1) will make you more understand what will be going on.
3. For API implementation, please follow the given spec below (Appendix #2). We will tell you a secret we will run your API through our awesome automated tests. However, we have a GraphQL version here https://github.com/pangaunn/election-api.

4. We hope you can figure out what are and how to “open the vote” and “close the vote.”
5. You are allowed to use any programming languages, frameworks, or tools to make the job done. Show off yourself to your best advantage.
6. When we review your code, we are also looking for;

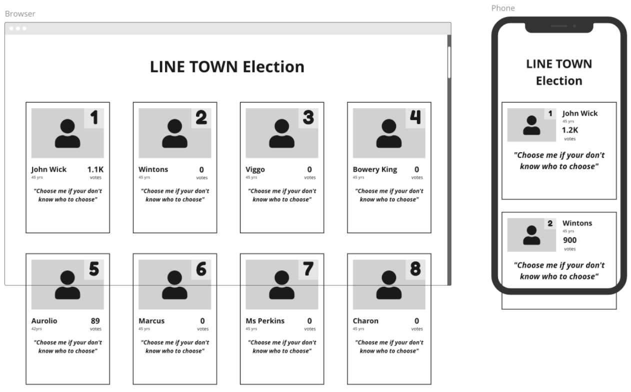
   Clarity: You can write clear code that any devs could read and understand in one go
   Simplicity: You can write gimmick-free and straightforward code with no ambiguities
   Defensiveness: You can cover edge cases and treat user inputs with care
   Resilience: You can gracefully handle an error and unexpected behavior

   Without these, your points may get a penalty.
7. Write a README to explain how to set up, install and run your system. And also, explain what you've done so far apart from the main requirements. Your system should be containerized and able to function in a single command. If not, don't forget to explain it more in the README.
8. This job may seem a lot to you, so do it as much as you can within the given time. However, please prioritize the main requirements first rather than other optional or fancy ones.
9. To submit your work, please use the link given in the email, compress what is necessary to a single file (e.g., .zip), and upload it. When we review your assignment we will extract the compressed file, read your README, run your work locally, and test it against all the requirements.

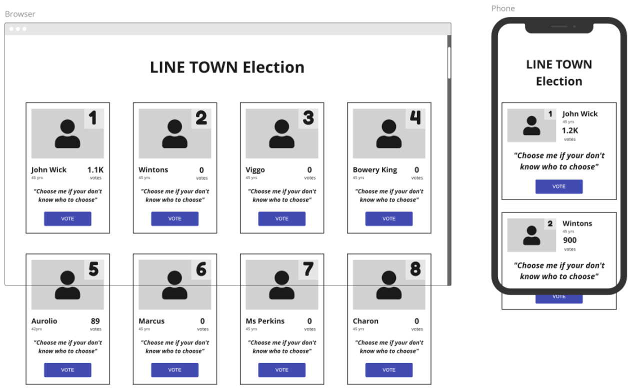
Appendix #1 - Wireframe

Wireframes

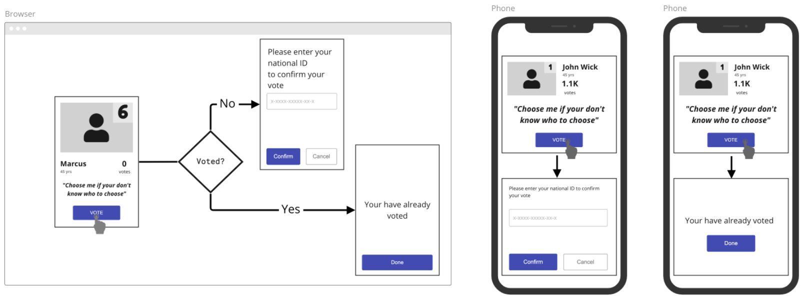
Candidate list - before open the vote


Browser
LINE TOWN Election
1 John Wick 1.1K votes "Choose me if your don't know who to choose"
2 Wintons 0 votes
3 Viggo 0 votes
4 Bowery King 0 votes
5 Aurolio 89 votes
6 Marcus 0 votes
7 Ms Perkins 0 votes
8 Charon 0 votes
Phone
LINE TOWN Election
1 John Wick 1.2K votes "Choose me if your don't know who to choose"
2 Wintons 900 votes "Choose me if your don't know who to choose"

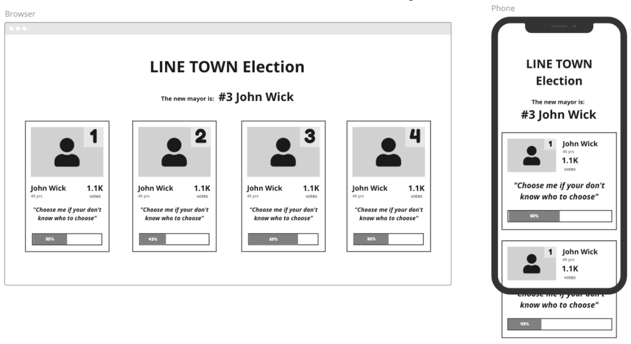
Candidate list - after open the vote



Vote Candidate


Browser
6 Marcus 0 votes VOTE
Voted?
No
Please enter your national ID to confirm your vote
Confirm
Cancel
Yes
Your have already voted
Done
Phone
1 John Wick 1.1K votes VOTE
Please enter your national ID to confirm your vote
Confirm
Cancel
Phone
1 John Wick 1.1K votes VOTE
Your have already voted
Done

Candidate list - after close the vote (Report)



Appendix #2 - API Spec

NOTES

This spec is designed to test candidate ability to design and implement backend system. Some might have security issues (and thus not production ready), please feel free to add security measures/improvements!

GET Candidate

API to get Candidate list

Example curl:

```
curl {endpoint}/api/candidates \
--header 'Authorization: Bearer xxxx'
```

Expected Response:

```
[
  {
    "id": "1",
    "name": "Elon Musk",
    "dob": "June 28, 1971",
    "bioLink": "<https://en.wikipedia.org/wiki/Elon_Musk>",
    "imageLink": "<https://upload.wikimedia.org/wikipedia/commons/e/ed/Elon_Musk_Royal
_Society.jpg>",
    "policy": "Lorem Ipsum is simply dummy text of the printing and typesetting indust
ry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when
an unknown",
    "votedCount": 0
  },
  {
    "id": "2",
    "name": "Jeff Bezos",
    "dob": "January 12, 1964",
    "bioLink": "<https://en.wikipedia.org/wiki/Jeff_Bezos>",
    "imageLink": "<https://pbs.twimg.com/profile_images/669103856106668033/UF3cgUk4_40
0x400.jpg>",
    "policy": "Lorem Ipsum is simply dummy text of the printing and typesetting indust
ry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when
an unknown",
    "votedCount": 0
  },
  ...
]
```

Response Detail

Parameter	Description
id	ID of the candidate
name	Candidate's name
dob	Candidate's Date of birth
bioLink	Candidate's biography link
imageLink	Candidate's image
policy	Candidate's policy
votedCount	Candidate's vote count

GET Candidate Detail

API to get Candidate detail

Example curl:

```
# curl {endpoint}/api/candidates/:candidateId \
# --header 'Authorization: Bearer xxxx'

# Example
curl {endpoint}/api/candidates/1 \
--header 'Authorization: Bearer xxxx'
```

Parameter Detail

Parameter	Description
candidateId	ID of the candidate

Expected Response:

```
  {
    "id": "1",
    "name": "Elon Musk",
    "dob": "June 28, 1971",
    "bioLink": "<https://en.wikipedia.org/wiki/Elon_Musk>",
    "imageLink": "<https://upload.wikimedia.org/wikipedia/commons/e/ed/Elon_Musk_Royal
_Society.jpg>",
    "policy": "Lorem Ipsum is simply dummy text of the printing and typesetting indust
ry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when
an unknown",
    "votedCount": 0
  }
```

Response Detail

Parameter	Description
id	ID of the candidate
name	Candidate's name
dob	Candidate's Date of birth
bioLink	Candidate's biography link
imageLink	Candidate's image
policy	Candidate's policy
votedCount	Candidate's vote count

POST Create a new Candidate

API to create a new Candidate

Example curl:

```
curl -X POST {endpoint}/api/candidates \
--header 'Authorization: Bearer xxxx' \
-d '{
  "name": "Brown",
  "dob": "August 8, 2011",
  "bioLink": "<https://line.fandom.com/wiki/Brown>",
  "imageLink": "<https://static.wikia.nocookie.net/line/images/b/bb/2015-brown.png/rev
ision/latest/scale-to-width-down/700?cb=20150808131630>",
  "policy": "Lorem Ipsum is simply dummy text of the printing and typesetting industr
y. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when
 an unknown"
}'
```

Parameter Detail

Parameter	Description
name	Candidate's name
dob	Candidate's Date of birth
bioLink	Candidate's biography link
imageLink	Candidate's image
policy	Candidate's policy

Expected Response:

```
  {
    "id": "3",
    "name": "Brown",
    "dob": "August 8, 2011",
    "bioLink": "<https://line.fandom.com/wiki/Brown>",
    "imageLink": "<https://static.wikia.nocookie.net/line/images/b/bb/2015-brown.png/r
evision/latest/scale-to-width-down/700?cb=20150808131630>",
    "policy": "Lorem Ipsum is simply dummy text of the printing and typesetting indust
ry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when
an unknown",
    "votedCount": 0
  }
```

Response Detail

Parameter	Description
id	ID of the candidate
name	Candidate's name
dob	Candidate's Date of birth
bioLink	Candidate's biography link
imageLink	Candidate's image
policy	Candidate's policy

PUT Update a Candidate

API to update a Candidate

Example curl:

```
curl -X PUT {endpoint}/api/candidates/3 \
--header 'Authorization: Bearer xxxx' \
-d '{
  "number": 3,
  "name": "LINE Brown",
  "dob": "August 8, 2011",
  "bioLink": "<https://line.fandom.com/wiki/Brown>",
  "imageLink": "<https://static.wikia.nocookie.net/line/images/b/bb/2015-brown.png/rev
ision/latest/scale-to-width-down/700?cb=20150808131630>",
  "policy": "Lorem Ipsum is simply dummy text of the printing and typesetting industr
y. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when
 an unknown"
}'
```

Parameter Detail

Parameter	Description
candidateId	ID of the candidate
name	Candidate's name
dob	Candidate's Date of birth
bioLink	Candidate's biography link
imageLink	Candidate's image
policy	Candidate's policy

Expected Response:

```
# Updated candidate's detail as response
  {
    "id": "3",
    "name": "LINE Brown",
    "dob": "August 8, 2011",
    "bioLink": "<https://line.fandom.com/wiki/Brown>",
    "imageLink": "<https://static.wikia.nocookie.net/line/images/b/bb/2015-brown.png/r
evision/latest/scale-to-width-down/700?cb=20150808131630>",
    "policy": "Lorem Ipsum is simply dummy text of the printing and typesetting indust
ry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when
an unknown",
    "votedCount": 0
  }
```

Response Detail

Parameter	Description
id	ID of the candidate
name	Candidate's name
dob	Candidate's Date of birth
bioLink	Candidate's biography link
imageLink	Candidate's image
policy	Candidate's policy

DELETE Delete a Candidate

API to delete a Candidate

Example curl:

```
curl -X DELETE {endpoint}/api/candidates/3 \
--header 'Authorization: Bearer xxxx'
```

Parameter Detail

Parameter	Description
candidateId	ID of the candidate

Expected Response:

```
# success
  {
    "status": "ok"
  }

# fail
  {
    "status": "error",
    "message": "Candidate not found"
  }
```

Response Detail

Parameter	Description
status	status of request
message	(optional) error message

POST Check Vote status

API to check vote status for voter

Example curl:

```
curl -X POST {endpoint}/api/vote/status \
--header 'Authorization: Bearer xxxx' \
--header 'Content-Type: application/json' \
-d '{
  "nationalId": "1111111111114"
}'
```

Expected Response:

```
# Success
{
  "status": true
}

# Error
{
  "status": false
}
```

Response Detail

Parameter	Description
status	Current vote status, if this national id can vote or not

POST Vote

API to vote

Example curl:

```
curl -X POST {endpoint}/api/vote \
--header 'Authorization: Bearer xxxx' \
--header 'Content-Type: application/json' \
-d '{
  "nationalId": "1111111111114",
  "candidateId": 1
}'
```

Expected Response:

```
# Success
{
  "status": "ok",
}

# Error
# In case, voter has already voted
{
  "status": "error",
  "message": "Already voted"
}

# In case, election is closed
{
  "status": "error",
  "message": "Election is closed"
}
```

Response Detail

Parameter	Description
status	status of request
message	(optional) error message

POST Toggle Election

Example curl:

```
curl -X POST {endpoint}/api/election/toggle \
--header 'Authorization: Bearer xxxx' \
--header 'Content-Type: application/json' \
-d '{
  "enable": true
}'
```

Expected Response:

```
# Success
{
  "status": "ok",
  "enable": <true|false>
}
```

Response Detail

Parameter	Description
status	status of request
enable	status of election (open or close as true/false)

GET Election Result

Example curl:

```
curl -X POST {endpoint}/api/election/toggle \
--header 'Authorization: Bearer xxxx' \
--header 'Content-Type: application/json' \
-d '{
  "enable": true
}'
```

Expected Response:

```
# Success
[
  {
    "id": "1",
    "votedCount": 10
  },
  ...
]
```

Response Detail

Parameter	Description
id	id of candidate
votedCount	how many votes did the candidate get

GET Election Result

Example curl:

```
curl -X GET {endpoint}/api/election/result \
--header 'Authorization: Bearer xxxx'
```

Expected Response:

```
[
  {
    "id": "1",
    "name": "Elon Musk",
    "dob": "June 28, 1971",
    "bioLink": "<https://en.wikipedia.org/wiki/Elon_Musk>",
    "imageLink": "<https://upload.wikimedia.org/wikipedia/commons/e/ed/Elon_Musk_Royal
_Society.jpg>",
    "policy": "Lorem Ipsum is simply dummy text of the printing and typesetting indust
ry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when
an unknown",
    "votedCount": 3,
    "percentage": "75%"
  },
  {
    "id": "2",
    "name": "Jeff Bezos",
    "dob": "January 12, 1964",
    "bioLink": "<https://en.wikipedia.org/wiki/Jeff_Bezos>",
    "imageLink": "<https://pbs.twimg.com/profile_images/669103856106668033/UF3cgUk4_40
0x400.jpg>",
    "policy": "Lorem Ipsum is simply dummy text of the printing and typesetting indust
ry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when
an unknown",
    "votedCount": 1,
    "percentage": "25%"
  }
]
```

Response Detail

Parameter	Description
id	ID of the candidate
name	Candidate's name
dob	Candidate's Date of birth
bioLink	Candidate's biography link
imageLink	Candidate's image
policy	Candidate's policy
votedCount	Candidate's vote count
percentage	Candidate's vote percentage

GET Exported Result (download)

Example curl:

```
curl -X GET {endpoint}/api/election/export \
--header 'Authorization: Bearer xxxx'
```

Expected Response:

CSV file download with columns:

- Candidate id
- National id

Real-time Vote Stream

Websocket stream for real-time vote count

Update Speed: real-time

Payload:

```
{
  "id": "1",     // Candidate ID
  "votedCount": 10      // Total number of votes for this particular candidate
}
```