COMP SCI 7412 Group Project Milestone 1 - Individual

Tinson Lai
a1812422

The voting system should have the following functional requirements:

1. User must be able to register into the system.
2. The eligibility of the user must be verified before participating the electoral rolls.
3. User must be able to login after registration.
4. User may be able to update their personal details including but not limited to permanent address and email address. Critical information may not be changed manually
5. The system must identify the type of user after login. Specifically,
    - An **elector** must be able to submit a ballot.
    - **AEC officers** must be able to organise an election in the election day of specified by the rules.
6. The ballot can only be submitted exactly once per electoral roll.
7. The system must strictly complies with the AEC rules and definitions.
8. The system must be able to yield a final statistic after the election. Preferably, it may provide a real-time statistical functionality during the open period.

Based on these functional requirements, it can be extended to include the following security requirements:

1. Password must be protected during communication with the server.
2. Eligibility verification must be backed by the databases of personal information held by both federal and state governments as well as their affiliated official organisations. The plan is to run a simulating RESTful server to complete the verification step. This architecture is still necessary even if the access to the actual personal identity databases is granted since this can minimise the risks of leaking personal information unintentionally and avoid potential privacy breaches caused by malicious attacks. Another advantage is that the RESTful APIs usually subject to less frequent changes once published regardless of the internal implementation details, so it is easy to implement various verification modules inside the RESTful server with a consistent output API model. It is also easier to extend the implementation to mobile platform in the future as this functionality has already been separated from the application implementation.
3. The system must have an access control system. The granularity of the system depends on the actual implementation, but the basic guidelines are:

- The elector should not be able to see ballots submitted by others. Meanwhile, according to the current electoral system, the elector will not be able to review or modify the ballot once submitted.
- The organisers should not be able to modify the contents of any ballots.

The technology stack for this project is:

- Frontend: A licensed ReactJS responsive web template with modification to fit the functional requirements. It's possible to introduce other frameworks such as Bootstrap into the implementation if needed. The whole server will have SSL enabled.
- Backend:
  - Firebase Authentication for user registration and login. Firebase Authentication is essentially a wrapper over a relational database, but the functionality it provides has already eliminate the possibility of any kinds of attacks targeting relational databases. The communication between devices and Firebase Authentication will also be encrypted.
  - Firebase Cloud Firestore for permanent data storage. Firebase Cloud Firestore is a NoSQL cloud database as a successor of Firebase Realtime Database. It has guaranteed availability with very sophisticated and flexible query system. It is also easier to aggregate the results to yield the statistics.
  - The RESTful server will be implemented with Spring Boot in Kotlin to provide the functionality specified above, and it includes a trivial verifier which will let all accounts pass through the verification for this project. JVM based Spring Boot project is a compromise between development efficiency and server performance. The server address will have SSL enabled so the communication will be encrypted. The current plan is to send only the user UUID generated by Firebase Authentication from the website to the RESTful server, and the server will respond once the verification step is completed. The server will read those personal information from and write the verification result to Firebase Cloud Firestore directly. The response from the server will be just a status code informing the client that the result is ready. There are several advantages of this design:
    * Firebase Authentication UUID is highly random and unique to the whole account database, and this will only be used as an internal identifier within the system. Even the account holder will not be able to know this ID.
    * Even if the UUID is indeed leaked by malicious attacks, a CSRF attack to the RESTful server will not disclose any further information.
    * Firebase is designed and implemented with CSRF protection.
- `https://group-project.sse.uoa.tinson.dev` for a demonstration of the website.
- `https://api.group-project.sse.uoa.tinson.dev` for an example of the corresponding RESTful API server.