PLANNING

1. Libraries & Frameworks:

A. Online Server / Backend:

- **Cloud Backend as a Service (BaaS):**

A rapidly growing cloud computing solution for tech enthusiasts and businesses seeking to save costs on building and maintaining their own backend infrastructure. A prominent player in the BaaS market is Google's Firebase product.

Firebase:

- Initially a real-time database, it has now grown to encompass 18 services (4 of which are in beta) and dedicated APIs.
- Offers a comprehensive Backend-as-a-Service solution for building, testing, and managing both mobile and web applications.
- **Reasons for choosing Firebase for the project:**
    - Suitable for both small-scale initial development and future scalability.
    - Secure and easy-to-implement platform.
    - Key features relevant to the project:
        - Realtime Database (Firestore)
        - Push Notifications
        - Authentication (Email & Password, Google, Facebook, Github)
        - Built-in Data Security with Node-Level Access Control
        - Stream-based Data Handling for Highly Scalable Applications

B. Desktop Client / Frontend:

1. Building Cross-Platform Applications with Electron and Node.js:

- **Node.js:**
    - The underlying platform for Electron, offering benefits such as a vast plugin ecosystem.
    - The npmJS package repository provides the world's largest collection of open-source libraries, further expanding development capabilities.

2. Streamlining UI Development with Bootstrap:

- **Bootstrap:**
    - A powerful toolkit composed of HTML, CSS, and JavaScript tools designed for building web pages and applications.
    - **Key benefits for the project:**

- Responsive Design: Ensures optimal user experience across various screen sizes.
- Extensive Browser Compatibility: Guarantees our application functions properly on a wide range of browsers.
- Consistent Design with Reusable Components: Promotes a cohesive user interface and simplifies development.
- Rich Extensibility with JavaScript and jQuery Support: Allows for customization and integration with existing JavaScript plugins.
- Ideal choice for front-end development in this project due to the lack of user-friendly tools for UI design and visual manipulation of HTML and CSS code.

2. Approaching method

A. System Architecture & Development Progress

a) From Requirements & User Stories to System Design

Three different types of requirements that describes requirements from three different points of view to the system.

- **Enumerated Functional Requirements** are requirements that specify expected system behavior & individual function/features of the system (Ex: R5 said that task can be setted start time and end time) and the development priority of that function. Higher priority weight means that the feature/ function will be implemented sooner and more important.
- **Enumerated Non-Functional Requirements** are requirements that specify other characteristics of the system that are not related to system function/features, such as server maintainability, security, user logging & scalability… (ExL R20 said that logging and monitoring of the system and application must be in place) From this type of requirements, need to design system architecture decisions in the following section.
- **User Interface Requirements** are requirements that specify user interface related requirements. It includes the visually view of end user and the way they interact with the system through an interface.

Between **CLI** (Command Line Interface) and **GUI** (Guided User Interface), as we targeted the general consumers who are mostly unexperienced & schools, not professionals, building a user-friendly **GUI** is a better approach

b. System Architecture Decision

The system architecture should be designed based on guidelines from the prototype of a simple to-do list (which is described at 1.1.A) with intention to scale it to a bigger, higher practicality TPM system.

Architectural decisions influence and impact the non-functional characteristics of a system.

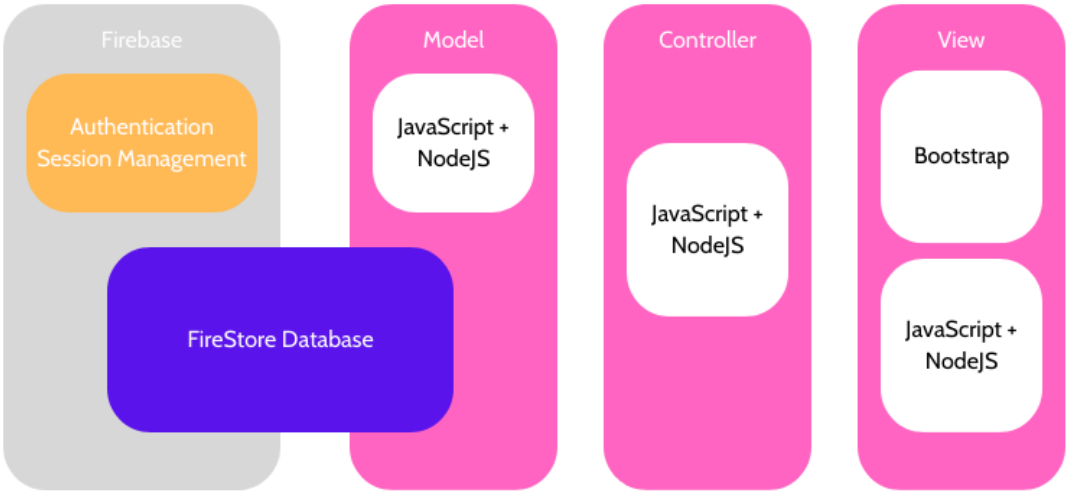
c) Implementing Technologies to System Architecture

Frontend (Vanilla JavaScript):

- **HTML, CSS, JavaScript**: These core web technologies will be used to build the user interface and client-side logic.
- **Model-View-Controller (MVC) Pattern**: Maintain the MVC pattern for structuring the application. The model holds election data and logic, the controller handles user interactions and interacts with the model, and the view renders the UI.
- **Bootstrap** : for pre-built UI components and utilities to simplify development.

Backend (Node.js, Firebase):

- **Node.js**: This remains the server-side runtime environment handling data processing and business logic.
- **Firebase**: Continues to function as the backend, providing user authentication, database, storage, and cloud functions for the election application.



d. Refactoring

Refactoring is a systematic process of improving code without creating new functionality that can transform a mess into clean code and simple design.
In a newly designed system, refactoring is almost unavoidable.
As OOP (Object-oriented Programming) rules and layer isolation is strictly applied on this project, the codebase is super easy to refactor.

e. Testing & Deploying

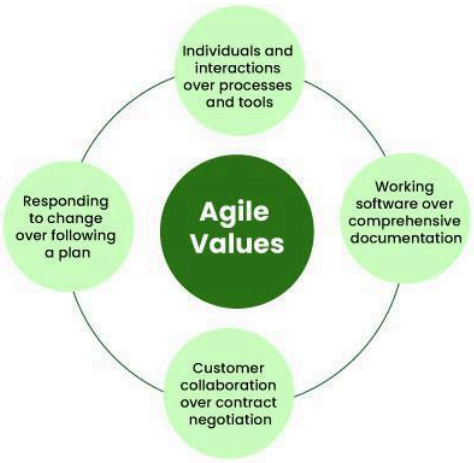
Software testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test.
In this project, the testing progress follows the standard lifecycle of software testing (diagram below)
Test cases are designed specifically for each sprint of the Scrum progress, and should be sorted to meet the highest priorities first

B. Project Management

Agile & Scrum

Agile Software Development is a software development methodology that values flexibility, collaboration, and customer satisfaction. It is based on the Agile Manifesto, a set of principles for software development that prioritize individuals and interactions, working software, customer collaboration, and responding to change. The Agile Software Development Methodology Manifesto describe four core values of Agile in software development:

1. Individuals and Interactions over Processes and Tools
2. Working Software over Comprehensive Documentation
3. Customer Collaboration over Contract Negotiation
4. Responding to Change over Following a Plan



4 Values of Agile

Scrum is an agile team collaboration framework commonly used in software development and other industries.

Scrum prescribes for teams to break work into goals to be completed within time-boxed iterations, called sprints. Each sprint is no longer than one month and commonly lasts two weeks. The scrum team assesses progress in time-boxed, stand-up meetings of up to 15 minutes, called daily scrums. At the end of the sprint, the team holds two further meetings: one sprint review to demonstrate the work for stakeholders and solicit feedback, and one internal sprint retrospective.