Projekt-Spezifikation

Projektname: Vokko Voting App

1. Thema

Abstimmungen und Wahlen in Vereinen, KMUs oder sonstigen kleineren Teams durchführen.

2. Umfeld, Ausgangslage

Die Projektidee entstand innerhalb eines anderen Studiengangs (MSc in Innovation and Entrepreneurship HEC Paris), den einer der beiden Studierenden (Remo Peduzzi) parallel besucht. Dort ist die Idee, eine universelle E-Voting App für diverse Einsatzbereiche zu kreieren. Das MVP fokussiert aber auf das Setup und die Durchführung einer Generalversammlung (General Meeting).

3. Aufgabenstellung & Ziel des Projektes

Wir haben uns entschieden, uns auf folgende Szenarien auszurichten:

- Interaktiv durchgeführte Veranstaltungen, also «Live»-Abstimmungen und -Wahlen, nicht solche, die sich über mehrere Tage oder Wochen erstrecken
- Kleinere Teams wie zum Beispiel Vereine oder KMUs.

Viele Institutionen wie zum Beispiel Vereine konnten während der Pandemie ihre Mitgliederversammlungen nicht wie gewohnt ausführen und mussten auf improvisierte Alternativen ausweichen, zum Beispiel Abstimmungen auf dem Postweg – mit dem Nachteil, dass vor dem Entscheid keine Diskussion stattfand und keine Fragen gestellt werden konnten. Umgekehrt konnten die Mitgliederversammlungen auch nicht einfach per Zoom durchgeführt werden, weil ein einfaches, vertrauenswürdiges Tool zum Durchführen der Abstimmungen fehlte.

Die E-Voting-App füllt diese Lücke. Damit kann man Versammlungen remote oder hybrid durchführen, so dass jedes mit einem Handy oder Browser bewehrte Mitglied bei den Abstimmungen und Wahlen teilnehmen kann und die Ergebnisse sehen kann.

Der Organisator kann im Voraus Wahlen bzw. Abstimmungsfragen erfassen.

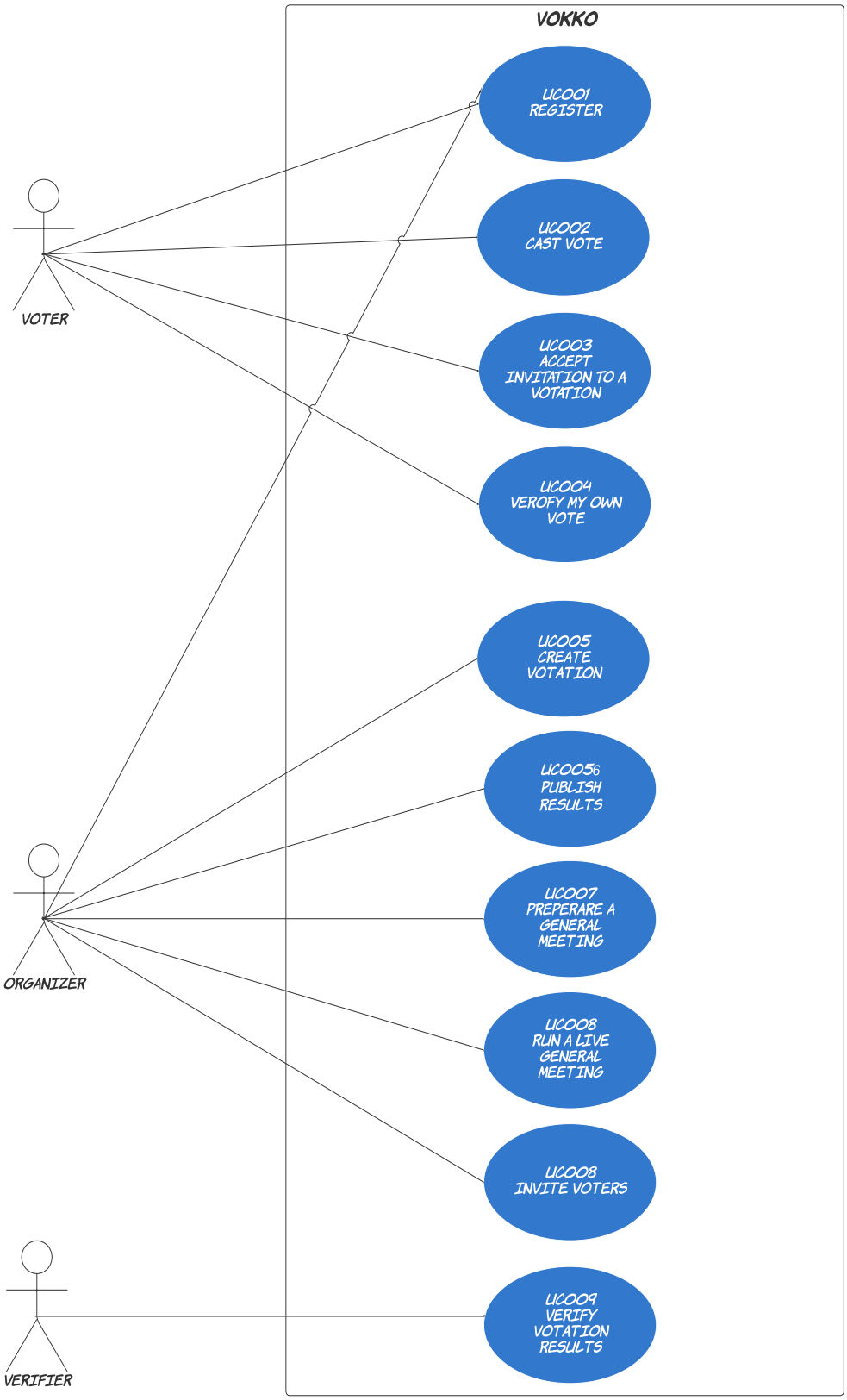
Mit Online-Kollaboration-Tools wie Mentimeter oder Miro kann man zwar auch Votings durchführen. Es fehlen aber Transparenz und Nachvollziehbarkeit.

4. Scope des Projektes

Die E-Voting-App soll folgende Features beinhalten:

- Erfassung von Abstimmungsfragen bzw. Wahlen mit Kandidaten
- Durchführung der Versammlung
    - Organisator-Sicht:
        - Der Organisator bestimmt Start, Dauer und Ende der Wahl
        - End-Resultatanzeige
    - Wähler-Sicht:
        - Stimmabgabe resp. Kandidatenauswahl
        - End-Resultatanzeige
- Nach der Versammlung: Reports + Charts über die einzelnen Entscheide (Ja, Nein, Enthaltungen etc.)
- Erweiterungsmöglichkeiten
    - Wähler verwalten
    - Kandidatenprofile pflegen
    - Laufende Resultatanzeige
    - Eigene Stimmabgabe überprüfen, Visualsierung der Blockchain
    - Rolle „Verifier“: Überprüfung der Resultate

4.1. Use Cases


VOKKO
UC001 REGISTER
UC002 CAST VOTE
UC003 ACCEPT INVITATION TO A VOTATION
UC004 VEROFY MY OWN VOTE
UC005 CREATE VOTATION
UC0056 PUBLISH RESULTS
UC007 PREPERARE A GENERAL MEETING
UC008 RUN A LIVE GENERAL MEETING
UC008 INVITE VOTERS
UC009 VERIFY VOTATION RESULTS
VOTER
ORGANIZER
VERIFIER

Use Case	UC001 Register	
Actor	Voter/ Organizer / Verifier	
Pre-Condition	None	
Post-Condition	User is registered	
Basic Path	**Actor**	**System**
	• User opens app. • App asks the user if he wants to restore from Google Drive. • User chooses not to. • User clicks on Registration Link/ Button • User puts in some personal data including a unique identifier such as email. • User chooses if he wants to back up his data on Google Drive • User fills in the Google Credentials and a Master Passcode	System creates keypair for the Users and saves it to the localStorage System registers the user on the public blockchain (public key and public identifiers only) System encrypts all the localStorage Data and saves it to the Google Drive

Use Case	UC002 CAST VOTE	
Actor	Voter	
Pre-Condition	• User is registered • User has been invited to the meeting • User has joined the meeting	
Post-Condition	User cannot vote for the specific voting anymore	
Basic Path	**Actor**	**System**
	• User receives notification of a starting vote • User choses an option • User confirms the option • User is redirected to the home screen	System registered that the user voted on the blockchain and registers the option chosen without any link.

Use Case	UC003 ACCEPT INVITATION TO A VOTING	
Actor	Voter	
Pre-Condition	• User is registered • User has been invited to the meeting	
Post-Condition	User is a registered voter for the voting	
Basic Path	**Actor**	**System**
	• User clicks on the email link	Browser is opened by the email client System registered that the user for the voting behind the scenes

Use Case	UC004 VERFIY ONE OF MY OWN VOTES	
Actor	Voter	
Pre-Condition	• User is registered • User has been invited to the meeting • User has joined the meeting	
Post-Condition	None	
Basic Path	**Actor**	**System**
	• User opens the app • The user navigates to the voting he wants to verify • User verifies the cote	The system verifies the block in the chain. The system does some fancy animation to illustrate verification. It turns either green or red depending on the outcome.

Use Case	UC005 CREATE A VOTING	
Actor	Organizer	
Pre-Condition	User is registered	
Post-Condition	None	
Basic Path	**Actor**	**System**
	• User opens the app • User navigates to the organizer's view • User creates a vote	

Use Case	UC006 PUBLISH RESULTS	
Actor	Organizer	
Pre-Condition	Meeting has ended	
Post-Condition	None	
Basic Path	**Actor**	**System**
	• User opens the app • User navigates to the organizer's view • User navigates to the voting • User publishes the results	System writes a block with the results. The block is visible to all invited voters at least or even open to the world.

Use Case	UC007 PREPARE A MEETING	
Actor	Organizer	
Pre-Condition	User is registered	
Post-Condition	None	
Basic Path	**Actor**	**System**
	• User opens the app • User navigates to the organizer's view • User creates a new general meeting • User defines the date and title general meeting • User adds motions (questions with options or candidates) • User adds voters (participants)	

Use Case	UC008 RUN A LIVE MEETING	
Actor	Organizer	
Pre-Condition	User is registered	
Post-Condition	None	
Basic Path	**Actor**	**System**
	• User opens the app • User navigates to the organizer's view • User navigates to the general meeting • User starts the meeting	

Use Case	UC009 INVITE VOTERS	
Actor	Organizer	
Pre-Condition	User is registered	
Post-Condition	None	
Basic Path	**Actor**	**System**
	• User opens the app • User navigates to the organizer's view • User invites voters to the voting	

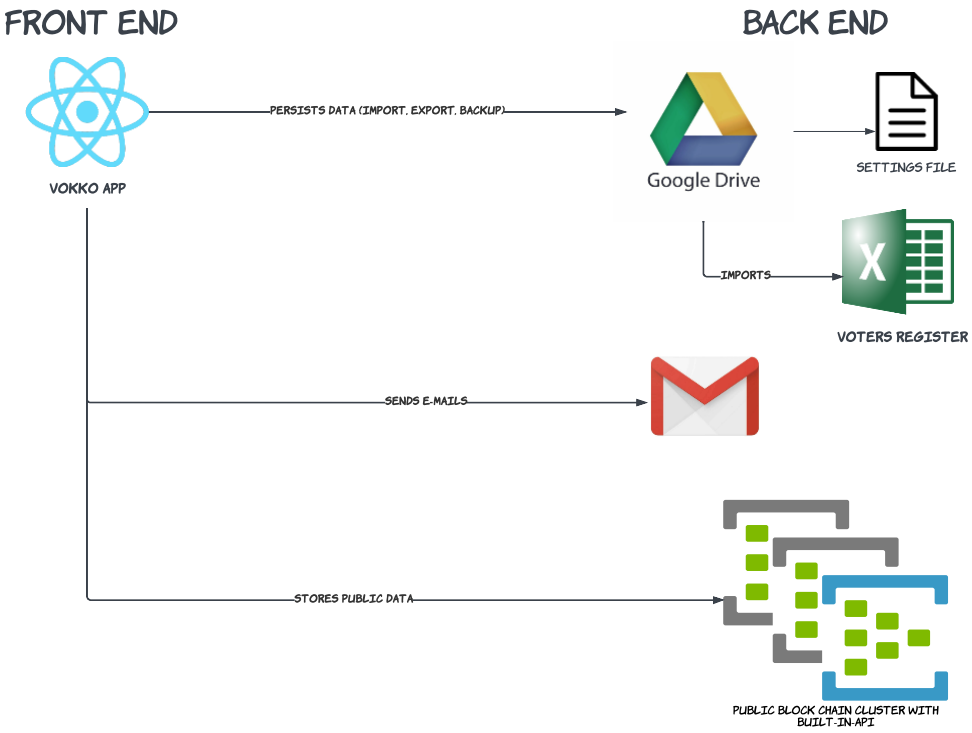
Use Case	UC010 VERIFY VOTING RESULTS	
Actor	Verifier	
Pre-Condition	User is registered, the voting ended, and the results were published	
Post-Condition	None	
Basic Path	**Actor**	**System**
	• User opens the app • User navigates to the verifier's view • User navigates to the voting to verify • The user verifies the voting	

4.2. Priorisierung

Use Case	Priority	Requirement Type
UC002 CAST VOTE	HIGH	MUST-HAVE
UC005 CREATE A VOTING	HIGH	MUST-HAVE
UC007 PREPARE A MEETING	HIGH	MUST-HAVE
UC006 PUBLISH RESULTS	HIGH	MUST-HAVE
UC008 RUN A LIVE MEETING	HIGH	NICE TO HAVE
UC009 INVITE VOTERS	MEDIUM	NICE TO HAVE
UC003 ACCEPT INVITATION TO A VOTING	MEDIUM	NICE TO HAVE
UC001 REGISTER	LOW	NICE TO HAVE
UC004 VERIFY ONE OF MY OWN VOTES	LOW	NICE TO HAVE
UC010 VERFIY RESULTS	LOW	WON'T IMPLEMENT

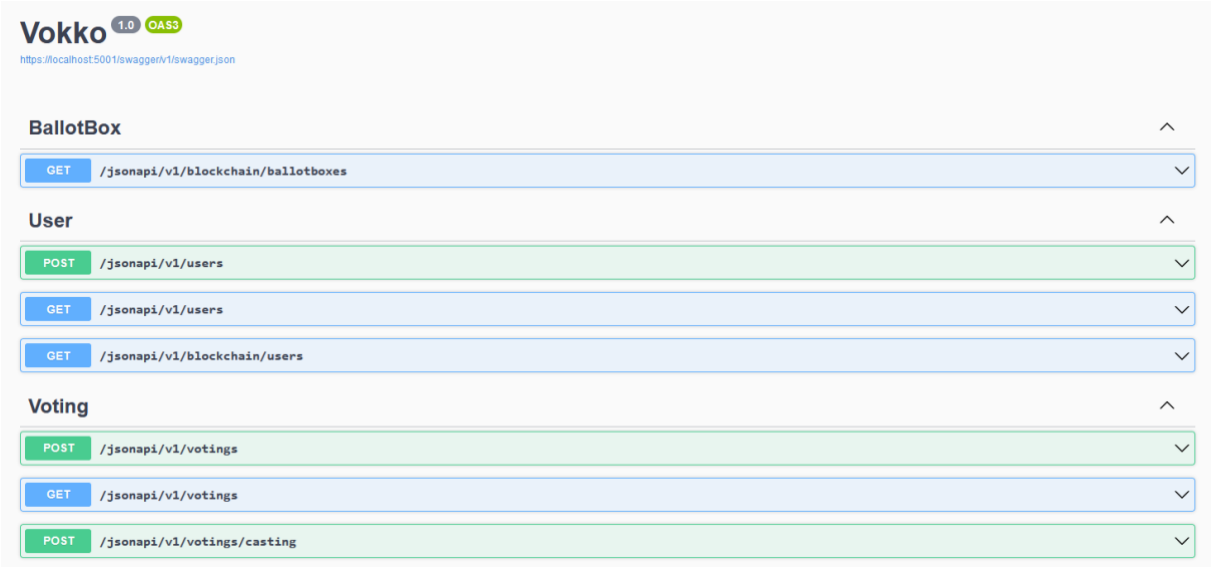
5. Grob-Architektur und Design

5.1. Architecture and System Context



5.2. API-Endpoints

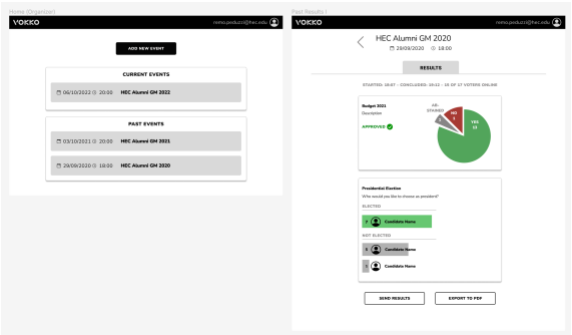
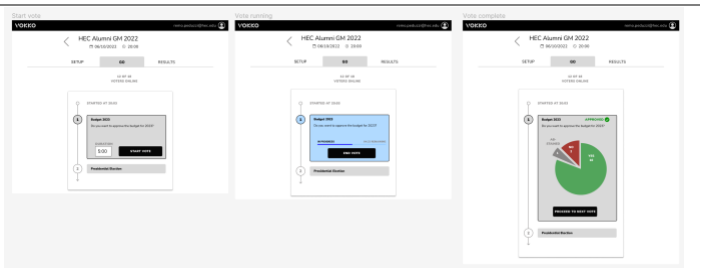
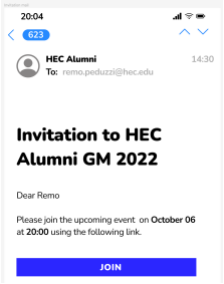
Im Moment existieren folgende Endpoints, die auch voll funktionieren. Es braucht sicher noch Endpoints für Generalversammlungen. Weitere werden falls nötig erstellt und wenn möglich werden Dinge auch nur gemockt.

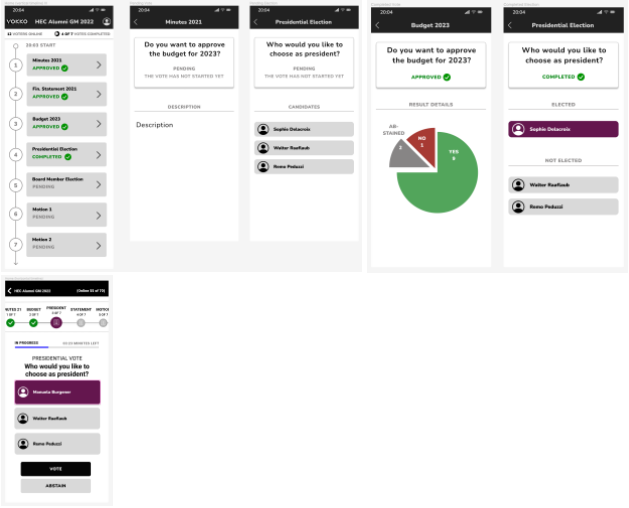


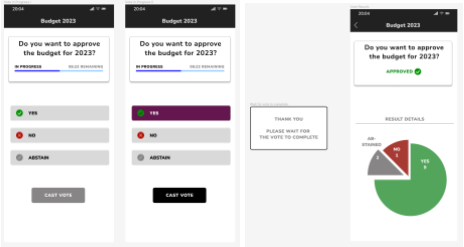
5.3. Überblick über die zu realisierenden Screens

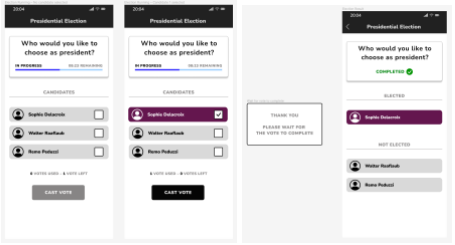
- Screens für die Organizer-View sind auf Desktop oder Tablet ausgerichtet
  - Meeting-Ablauf wird durch eine Timeline dominiert
  - Zu untersuchende Varianten: vertikale vs. Horizontale Timeline
  - Responsive Design, aber Optimierung für kleine Displays erst in zweiter Priorität
  - Die Organizer-Views sind bewusst eher einfach gehalten, z.T. im Formular-Stil
  - Der Versammlungs-Ablauf wird durch den Organizer gesteuert
    - Abstimmungen und Wahlen haben eine im Voraus bestimmte Zeitdauer
    - Können aber auch manuell beendet werden
  - Meeting-Setup und Meeting-Durchführung haben unterschiedliche, auf den Flow angepasste Darstellung derselben Daten
    - Zweite Prio: Änderung des Setups bei laufendem Meeting (zB Anträge)

- Screens für die Voter-View sind auf Mobile ausgerichtet
  - Meeting-Ablauf wird durch eine Timeline dominiert
  - Zu untersuchende Varianten: vertikale vs. Horizontale Timeline
  - Das UI soll den Voter möglichst nicht verwirren, daher Fokus auf allgemein verständliche UI-Idiome (Scrolling, Button, Checkbox)
  - Der Versammlungs-Ablauf wird durch den Organizer gesteuert
    - Der Voter folgt diesem Ablauf
    - Ausserhalb der eigentlichen Abstimmungen/Wahlen kann er durch die Vorlagen browsen

Actor	Flow	Aktionen	
Organizer	**Meeting-Übersicht und Resultate**	• Durch aktuelle und vergangene Meetings browsen • Reports	
	Meeting-Setup	• Neues Meeting • Titel und Datum • Vorlagen • Teilnehmer • Einladungen senden	
	Meeting durch-führen	• Meeting starten • Meeting beenden • Variante: horizontale Timeline	
	Abstim-mung	• Abstimmung starten • Abstimmung beenden • Resultate	
	Wahlen	• Wahl starten • Wahl beenden Resultate	
Voter	**Einladung**	• Per E-Mail-Link teilnehmen	
	Registrie-rung		

Übersicht	• Durch die Abstimmungs-/ Wahlvorlagen browsen • Kandidaten ansehen • Bisherige Resultate ansehen • Variante: horizontale Timeline	

Abstim-men	• Sachentscheide
	• Optionen:
	  ○ Ja
	  ○ Nein
	• Enthaltung
		

Wahlen	• *n* aus *m* Kandidaten wählen	

5.4. Technology Choice

Die Idee war zunächst, das Cross-Plattform Mobile App Framework Ionic einzusetzen, um gewisse Daten lokal auf dem Geräte persistieren zu können und somit eine dezentrale Architektur zu ermöglichen. Wir haben uns aber dazu entschieden in einer ersten Version eine PWA zu bauen, welche auch auf einem Computer verwendet werden kann. Da Remo Peduzzi in der Firma React.JS als SPA einsetzt haben wir uns aus Synergiegründen dafür entschieden dieselbe Technologie für das Projekt zu verwenden.

5.5. State-Management

Damit die Applikation später aus Sicherheitsüberlegungen auch dezentral eingesetzt werden kann, werden wir so ziemlich alles auch Client-seitig persistieren. Damit man beim Leeren des Cache nicht alle Daten verliert werden wir ein Backup der Daten auf Google Drive erstellen (So wie das WhatsApp z.B. macht.). Die Applikation soll soweit es geht auch Offline-Fähig sein.

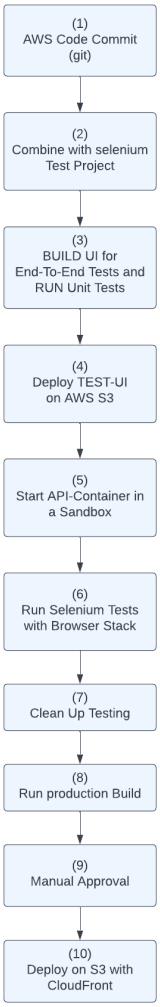
5.6. 3rd-Party API's and Components

Folgende 3rd Party Komponenten planen wir zu verwenden:

- Google Drive API
- Gmail API
- Material UI

5.7. Continuous Development and Integration

Wir werden bestehende Cloud Formation Templates verwenden um die CI/CD-Pipelines für das Frontend und Back-End in einem neuen Amazon AWS Account zu erstellen.



6. Studierende:

- Remo Peduzzi
- Walter Raaflaub