**Issues identified in background literature**

**Comprehension of New Concepts**

Voters often struggle to understand novel anti-coercion mechanisms without clear guidance. In a usability test of a coercion-resistant voting app (“Vote App”), many participants failed to grasp core features when instructions were minimal. For example, users did not realize that the system assigns a unique voting PIN that remains constant, nor that they were allowed to **re-vote multiple times** – concepts very different from typical apps. Several participants also misunderstood the purpose of the *fake credential (ruse PIN)* meant to thwart coercers, with some questioning why they needed a special “decoy” PIN at all when they could “simply enter a random one”​ [1]. These gaps in understanding highlight a challenge: if Loki’s UI does not explain its coercion-countering features clearly, voters may misuse them or fail to use them, undermining both usability and security.

**Memory Burden and Secrecy**

Many coercion-resistance strategies rely on the voter memorizing a secret or performing an extra step. This raises a **usability vs. security tension**: the voter must remember secrets (like an alternate password, PIN, or code) and recall them under stress, but humans have limited memory reliability. An interview study (N=26) on voter coercion found *memorability* to be a pivotal factor – most counterstrategies require the voter to remember a secret, and users perceived this as a usability hurdle​ [2]. If a voting system requires voters to retain a PIN or credential without writing it down (to avoid coercers finding it), memory lapses could lead to mistakes. Indeed, a user study of a fake-credential scheme showed ~10% of participants *mistakenly used the wrong credential to vote*, effectively invalidating their ballot​. This error occurred even though 96% of participants *understood* the concept of fake credentials in theory​ [3], underscoring how remembering and correctly using secrets under real conditions is challenging. Loki’s UI’s design must minimize memory load (or provide safe memory aids) to prevent user errors that could compromise their vote’s validity or privacy.

**Information Overload vs. Lack of Guidance**

Striking the right amount of instructional information is difficult. Voters need to be informed about unusual steps (like “enter a backup code” or “choose a control number”) but excessive text can overwhelm or annoy them. The Vote App study found that some participants felt the on-screen explanations were *too long or detailed*, while others felt those explanations were *essential* for such a security-critical application​. Too little information led to user confusion in that study – for example, encountering a screen with **“control numbers”** to select left many users asking *“What am I supposed to do now?”*​. At the same time, lengthy warnings or technical jargon can reduce usability. Loki’s UI needs to present guidance in a **concise, user-friendly manner**, perhaps with the ability to get more details on demand, to accommodate both user types. Finding this balance is an acknowledged challenge [1].

**User Perception of Need and Trust**

Another challenge is that not all voters perceive coercion as a real threat, which affects their willingness to tolerate extra security steps. Interviews reveal that if voters think voter coercion “could never happen here,” they may see anti-coercion features as unnecessary complications [2]. On the other hand, participants with personal or direct experience of coercion were more willing to use safeguards and even rated a coercion-resistant system as *equally trustworthy as traditional voting*​ [3]. This disparity means Loki’s UI must be approachable for the average voter who isn’t initially concerned about coercion, while still providing robust protections for those who need it. Building trust is key – if the UI appears too complicated or “paranoid,” some users might abandon it or distrust the system. Prior research noted that introducing entirely new voting concepts requires careful user education and even public awareness campaigns, but without undermining confidence (for example, scaring users about threats to the point they lose trust in the system)​ [3][4]. Essentially, Loki’s interface should normalize the security features as part of a modern voting experience, so that even sceptical users find it reasonable and not intimidating.

**Stress and Performance**

Finally, coercion scenarios are high-stress situations for users. A voter under pressure (real or perceived) is more prone to mistakes. Usability testing in controlled settings can only partially simulate this. For instance, researchers caution that it’s hard to truly assess how users will behave “in a situation which involves fear” [1]. Loki’s UI should therefore be as simple and foolproof as possible, because in a coercive moment a voter won’t have time to read help pages or recover from error. Any step that is confusing or easy to do incorrectly could result in the voter either complying with the coercer (failing the security) or accidentally invalidating their ballot while trying to resist. This is a **human-factor challenge** beyond normal usability: the design must be robust against stress-induced errors. For example, if a coerced voter needs to cast a fake vote, the UI should make that process discreet and hard to mess up – and similarly, casting a real vote afterwards (or before) should be straightforward and clearly indicated. Limited or no feedback (often a deliberate design choice to avoid giving the voter a telltale receipt) further increases the burden on the user to execute everything correctly in one go​ [4]. In short, Loki’s frontend must account for the *worst-case* user state (distracted, scared, being watched) and still enable a successful, private voting experience.

**Implications on UI design**

**Explain Security Features in Plain Language**

The UI must provide clear, succinct explanations for any non-standard voting step. Users should never be left guessing “why am I being asked to do this?” In Loki’s case, the user must identify their last previously cast valid ballot to change their vote. Here, a one-line prompt about its purpose can help (“Please select the ballot that matches your previous voting choices. This step helps ensure the security of your vote.”). The Vote App evaluation showed that lack of upfront explanation led to confusion about core functions​. At the same time, avoid alarming or technical language – instructions should be neutral and reassuring. Designers found it helpful to **remove exclamation marks or words like “ATTENTION!”** which can unnecessarily panic users. The UI must strive for a calm, guided experience where security steps feel like a normal part of the flow. [1]

**Use Progressive Disclosure for Help**

A practical way to balance information is to layer it. Loki’s UI should provide on-demand help (tooltips, “Learn more” pop-ups) rather than wall-of-text descriptions on every screen. Vote App’s [1] interface added a “magnifying lens” icon on almost every screen which users could tap to get more detailed info about that step​. The UI could similarly include context-sensitive help – for instance, a help icon next to the *re-vote* button explaining when and why to use it. This way, novice or uncertain users can easily access guidance, while experienced users aren’t slowed down. An **integrated user manual or FAQ** accessible from the interface is also recommended​  
[1]. By designing help as an optional layer, one can ensure voters have the information they need without overwhelming everyone by default.

**Provide Memory Aids (Securely)**

Given the importance of memorability, the UI should incorporate features that help users manage the required secrets or codes. One approach is using **recognition over recall** – for example, representing confirmation codes or PINs with easily recognizable images or emoji sequences in addition to text. Vote App displayed a string of “Private PIN Emojis” alongside the user’s PIN as a visual cue, which some users found helpful (familiar from apps like Telegram)​ [1]. Other research similarly found that user-generated visual markings can significantly improve retention of secrets without external notes​ [3]. Another aid is letting users **double-check or confirm entries**: for instance, a participant suggested allowing voters to enter their PIN twice to be sure they didn’t mistype it, since the second entry would catch if they mis-remembered it and saw the wrong emoji feedback [1]. However, any such aid must be covert if an adversary is present. Designers should avoid solutions that rely on visible notes or additional devices during the voting session, because a shoulder-surfing coercer could see those​. Instead, *the* *secret itself should be easier to remember* (e.g. a passphrase of four common words) and users should be guidance on memorization techniques upfront​ [4].

**Prevent Critical User Errors**

The UI should be designed to **minimise the risk of mistakes that compromise the vote or the secrecy.** A salient example is ensuring voters do not accidentally cast a **fake vote** when they intend a real vote. In the fake-credential study, 10% of users accidentally used their decoy credential to vote​ [3]. In Loki’s case, the user can avoid dealing with the security features when casting their first vote, but not when revoting. The UI should therefore provide enough information during the revoting process to minimise the risk of accidentally invalidate a vote meant to be valid. It should also inform the user about the possibility to vote physically in a situation where the user can’t recall their secrets.

**Normalize Coercion-Resistance Features for All Users**

A crucial lesson is to design the UI so that using anti-coercion features doesn’t itself look unusual. **Everyone using Loki should follow roughly the same visible process**, whether or not they are under duress, so that a coercer observing cannot distinguish a voter who is invoking a special countermeasure​. For example, if the system supports re-voting, the interface should encourage all voters to review and even recast their vote if they change their mind – not frame it as a niche “coercion recovery” feature. Estonia’s i-voting system did this by allowing any voter to vote multiple times (only the last vote counts), making re-voting a normal behaviour​ [4]. Loki can adopt a similar approach: advertise revoting as a convenience (“You can adjust your vote until the deadline”) so that a coerced voter’s extra vote doesn’t raise flags. The key is that nothing in the interface blatantly says “coercion mode” or otherwise tips off an observer. Blending these security features into the routine flow is one of the strongest lessons from the literature on coercion-resistant voting schemes [4].

**User-Friendly Verification Mechanisms**

End-to-end verifiability is often at odds with coercion-resistance, but many systems include some verification step (like checking a bulletin board for your vote’s hash or a confirmation code) that voters *can* perform. Designing this for usability is critical, otherwise few will do it or do it correctly. Studies on comparing cryptographic fingerprints offer guidance on presenting codes in human-friendly ways. They found that **graphical representations** (like patterns or avatar images) can make comparison quick and intuitive, but some formats led users to miss subtle differences, allowing attacks in tests​. A seemingly user-friendly method where participants had to “compare and select” the correct fingerprint from options performed *poorly* – many failed to notice when the correct option wasn’t present, so this approach is *not recommended* for critical verifications​. Instead, textual representations – even the “nerdy” hexadecimal strings – fared surprisingly well in usability and error detection​ [5]. The lesson for Loki is to choose a verification UI that maximizes accuracy for the average user. For instance, showing a short list of dictionary words or a two-row code (as some messaging apps do) might be easier to read and compare than one long hex string or a complex image. If Loki uses **emoji or symbols** as part of verification (as Vote App did), it should ensure users can easily tell if something is off. The Vote App experience showed that while some users liked emoji confirmations, others found them confusing or even thought the emojis were software glitches​ [1]. Thus, any novel verification aid (colours, animals, unicorn avatars, etc.) should be tested for clarity. One practical design might be combining modalities: e.g., show a small image *and* a short code – if they match what’s on the official site, great; if not, the user should know to report a problem. Ultimately, the verification step in Loki must be **as simple as “look and check”** with minimal mental calculation. If a voter has to manually compute something (even as simple as XORing numbers or summing values), the chance of error goes up dramatically​ [4]. Wherever possible, offload the complexity to the system and let the user just confirm a visible matching piece of data.

**Emphasize Security Benefits without Sacrificing UX**

The research suggests that users do appreciate security **when it’s made usable**. In the Vote App study, participants reported high satisfaction and specifically **appreciated the system’s security features** and overall ease of use​ [1]. This indicates that a well-designed UI can turn security measures into a net positive part of the user experience (users feel safer and in control, rather than burdened). Loki’s interface should frame features like revoting, verification, or credential checks as empowering options for the voter. For example, telling users “You can double-check that your vote was recorded correctly with this code – for your peace of mind” gives a positive spin. However, a caution is warranted: another study found that when users were given a stronger **security briefing** (making them more vigilant for threats), their subjective usability ratings went down slightly​. Essentially, reminding users of security risks can cause anxiety, affecting their comfort. The trade-off was that those users were far better at detecting a simulated attack (catching a fake voting terminal)​ [3]. The lesson for Loki is to find a balance in messaging – **educate users about security in a gentle, non-alarming way**. Perhaps include a short optional tutorial or use tooltips that mention security tips, rather than a scary warning banner. By carefully tuning the tone and amount of security messaging, Loki’s UI can keep users alert *without* overwhelming them. As one recommendation puts it, incorporate user education and awareness early, but ensure the additional information “will not overwhelm the voter or make them distrust the system”​ [4]. Achieving this balance is tricky but vital for a positive user experience.

**Iterative User-Centred Design**

A meta-lesson from all these papers is the importance of **testing with real users and refining**. Usable security doesn’t come for free – it requires iterative design. Kulyk and Neumann’s review explicitly recommends involving users from the start in developing coercion-resistant systems and getting feedback on prototypes​

[4]. Usability testing should be focused on Loki’s unique features (e.g., have users attempt to vote with and without a coercion scenario) to catch issues early. As seen in these studies, even well-intentioned designs had unforeseen pitfalls: e.g., users misinterpreting an icon, or failing to notice a crucial detail. By observing users, it can be discovered where the UI might be too confusing or too obvious (potentially revealing a covert action). Continuous refinement will significantly improve both usability and security. Moreover, providing **continuous support throughout the election** (help desks, hotlines, or in-app support chat) can mitigate the limited feedback inherent in coercion-resistant systems ​[4]. If a voter is unsure or encounters a problem, there should be resources to assist them (without the coercer’s knowledge if possible). In summary, building Loki’s frontend should be an iterative, user-informed process, not a one-off design – this ensures the final product is intuitive for voters from diverse backgrounds and capable of guiding them through a safe voting experience.

[1] Cristiano, L., Longo, R., & Spadafora, C. (2025). *Click and Cast: Assessing the Usability of Vote App*. Fondazione Bruno Kessler & University of Trento.

[2] Christina Nissen, Tobias Hilt, Melanie Volkamer, Jurlind Budurushi, and Oksana Kulyk. (2025). *Voting Under Pressure: Perceptions of Coercion and Counter-Strategies in Internet Voting*. USENIX Symposium on Usable Privacy and Security (SOUPS) 2025, Seattle, WA, United States.

[3] Merino, L.-H., Azhir, A., Zhang, H., Colombo, S., Tellenbach, B., Estrada-Galiñanes, V., & Ford, B. (2024). *E-Vote Your Conscience: Perceptions of Coercion and Vote Buying, and the Usability of Fake Credentials in Online Voting*. 45th IEEE Symposium on Security and Privacy.

[4] Kulyk, O., & Neumann, S. (2025). *Human Factors in Coercion-Resistant Internet Voting: A Review of Existing Solutions and Open Challenges*. IT University of Copenhagen & Independent Researcher.

[5] Tan, J., Bauer, L., Bonneau, J., Cranor, L. F., Thomas, J., & Ur, B. (2017). *Can Unicorns Help Users Compare Crypto Key Fingerprints?* Proceedings of the CHI Conference on Human Factors in Computing Systems, 3787–3799. <https://doi.org/10.1145/3025453.3025733>

WIP notes:

**Potential adversary tactics:**

The adversary forces the user to vote at the beginning of the voting period.  
Countermeasure: The user can revote later.

The adversary forces the user to vote at the very end of the voting period.  
Countermeasure: The user votes earlier and then provides false information during the revoting process, i.e. says they hadn’t voted before or intentionally misidentifies their previous ballots to invalidate the new vote.

The adversary forces the user to login at the beginning and makes them leave while the adversary submits a vote. Then, at the end of the process, the adversary once again forces the user to submit a vote.  
Countermeasure: There should be a way for one to vote in person and permanently lock their vote from being overridden.

**Potential memory aids for ballot identification:**

Timestamps – can be difficult to remember but also have the advantage of the user being able to recall them even in the event of the adversary forcing the user to login at the beginning of the voting period and making them leave the room while the adversary casts a vote for the user. Verification codes or symbols prevent the user from nullifying this vote. Vulnerable to the coercer knowing when the user last cast their valid vote.

Identicons – mentioned as confusing and not a bad idea in the context of page integrity verification

Emojis – mentioned as seemingly popular with some users while seeming unprofessional or even like glitches to others

Hex codes – mentioned as successful in the context of page integrity verification, testing should be done to determine their usefulness as ballot memory aids

Colours – personally seems like a good choice to use in combination with any of the above

**Usability testing logs:**

4th March w/ Ida – identified no issues, didn’t look through tutorials, said everything seemed intuitive -> the issue was that we did not explore the scenario of the coercer knowing she’s voted before prior to me explaining the system, I should provide no information and force the user to figure everything out on their own using the UI if I want to identify issues with the UI

4th March w/ Søren – quickly clicked though everything, again because everything seemed “obvious”, didn’t realise he submitted an invalid ballot during the coerced scenario -> it needs to be clearly and visibly communicated that this might happen

5th March w/ Adrian – lots of feedback about small things in the UI, wording, etc., mostly fixed now