Thunderbird Mobile - 2025 Product Strategy

👁️Vision statement

⭐North Star

- Mobile Mission: **"Fiddle less with your phone"**
- Vision: Design a mobile email platform that enhances user efficiency through focused triage, seamless desktop integration, and a suite of productivity-enhancing services.

A high percentage of people appear to be reading email on mobile first, or even making their mobile phone their primary device for reading email. This is especially true for the younger generation. On the other hand, work email is 40% less likely to be checked from a smartphone than personal email.

We need to find our niche where users are excited enough that they are willing to switch away from the default.

🗺️2025 Destination

We believe in the following high level goals:

- Release a testing version of Thunderbird for iOS that allows you to read and send email.
- Increase trust in emails being delivered to your inbox by lowering the negative reviews in this category to X%
- Introduce at least two new mechanisms that allow a user to focus on specific aspects of their email
- ~~Integrate with at least one of Thunderbird's Pro service offerings~~
- ~~Successfully launch an experiment that makes use of AI to increase email productivity~~
- Complete and document at least four key architectural refactorings aimed at simplifying code and reducing technical debt.

👫Target Customers

✅DOs	❌DON'Ts
We're competing with Outlook and Gmail. Focus on improvements that make customers more productive and allow them to challenge the defaults.	Focus solely on the power users and open source privacy enthusiasts. Compete only with other open source email clients.

These are some of the users we should focus our attention on:

- Appeal to mainstream users by emphasizing productivity
- Capture power users by going beyond the default
- Appeal to privacy focused users by providing encryption

We acknowledge that the target customers described here are quite opposite to Desktop's strategy. We believe this is still the right action to take because:

- Desktop and Android have a completely different codebase, form factor and ecosystem. We have the opportunity to take a different direction.
- Thunderbird Desktop already has a number of mainstream users, branching out into the niches allows us to acquire new groups of users.
- Thunderbird for Android has a very privacy-focused history with many niche features, which means we can bring in new users by heading more towards the mainstream users.
- Customizability is more difficult to achieve with the limitations of mobile operating systems, therefore we don't need to be as versatile as a Desktop application might need to be.

🌄Customers Horizons

General attitude and the values that dictate what we work on.

- **Make users spend less time on their phone**
  We're not trying to gamify reading email and get users hooked to increase active app time. We're trying to create the best email client for mobile. Our customers want to enjoy Thunderbird as a utility that helps them be productive and succeed.
- **When optimizing, optimize for reducing time spent between reading or writing email**
  We can't meaningfully change the time people spend reading or writing their email, but we can change how much time they spend with unproductive tasks, such as finding the right setting, applying a filter, finding the email they need.

📦Product Goals

✅DOs	❌DON'Ts
Instill confidence in users that our product will keep their email safe. Take lead in improving the funding experience and be a role model in MZLA. Ship user journeys that leave little room for dead ends. Make decisions based on data and put thought into what questions we should be asking. Make common tasks in email productivity quick and without extra steps.	Get lost in the details. Focus solely on stability and technical debt. Keep features or dive into edge cases because a small number of users have vocal needs.

👨‍🚀Engineering focus

- Every cycle we have to deliver value that users can see.
- Address technical debt in relevant areas tied to the target features before beginning work on those features.
- Develop and document sustainable team processes that are adaptable and accessible to all contributors.
- Maintain exceptional technical quality and rigorous testing practices for critical pathways, such as the funding experience.

- Ensure new code features adhere to modular design and follow current best practices

🎨 Design focus

- Larger changes should be spec'd by the design team. As a (senior) engineer, document how the feature works currently and what the target is, so design is able to hit the spot without much back and forth.
- Medium design changes can be developed up front, but should be validated with the design team.
- Smaller changes should be made with best effort and do not need a review from the design team.
- When making changes to the interface, think about the user journey from the previous version.
    - How much does muscle memory have to change?
    - Does someone unaware of the internals understand what the new UX does?
    - Is the language simple enough for someone without a technical background to understand?
- It is ok to change something and then change it again later, as long as this doesn't result in data loss for the user.

🏛️Priorities

There are a few pillars of existence we should use to categorize what we're doing:

- **Window Paint**: Thunderbird for Mobile needs to have a visual design that aligns with Thunderbird's identity. Users should feel empowered to effectively achieve their tasks.
- **Continuity**: Users should feel continuity when switching from Desktop to Mobile and back, so that it really feels like they are taking their communications with them when they leave their desk or couch.
- **Exclusivity (Android Only)**: We should use Thunderbird for Android as a vehicle to bring more exciting changes to Thunderbird exclusively. This way users are encouraged to switch over their app, and we can monitor adoption.
- **Productivity**: Being a leading email client is not only about maintaining status quo or making sure all features work, but taking the lead on productivity and communication on a mobile device.
- **Foundations and Parity**: Not every cool new platform feature needs to be a priority, but compatibility and the ability to ship is key. We also live off a sizable portion of technical debt that needs more than just a quick look while building new features.

Feature	Priority	Quarter	Pillar	Why	What	Tracking Goal *What goal are we making progress against?*
Funding Experience	P2	Q1 Q2 Q3 Q4	👩‍⚖️Foundations	Funding is a core aspect of how we succeed. We need to evolve the funding experience on a constant basis to ensure it remains effective	• Experiment with different approaches on signaling to the user we need support • Evolve the messaging to ensure it stays relevant • Develop a regular cadence that entices the user without nagging •	Foundational Work
Account Drawer Improvements	P1	Q1	🎨 Window Paint	A major release feedback item. It needs to be apparent to users how to switch to the account they want to go to. Different types of users have different needs in	• Add the ability to use icons to indicate different accounts. • A wide format that allows users to see the name and associated email of their account	Release Feedback

				presentation	• Hierarchical folder view	
Notifications System Overhaul	P1 High risk	Q1	Foundations	A major release feedback item is that emails are not arriving reliably. In part, this is because users don't have the right permissions enabled.	• Investigate scope for the changes. • Inform the user reliably if they have not provided the permissions needed for a seamless experience • Simplify the folder and account notification options, so that users have a clear	Increase trust in emails being delivered
iOS Architecture	P1	Q1	Foundations	Foundations for iOS	• General architecture on how components align • Decisions on code sharing • Create a backend for storing email content, user accounts, etc.	Release a testing version of Thunderbird for iOS that allows you to read and send email.
Android Platform Foundations	P2	Q2 Q3	Foundations		• Edge-to-edge support	Foundational Work
iOS Email Handling & Core	P1	Q2	Foundations	Foundations for iOS	• IMAP and extensible protocols support • Backend support for common inbox actions: delete, read/unread, etc	Release a testing version of Thunderbird for iOS that allows you to read and send email.
iOS Email Display	P2	Q2	Window Paint	Foundations for iOS	• Basic email display and message actions	Release a testing version of Thunderbird for iOS that allows you to read and send email.
~~Integrate Thunderbird Send~~	~~P1~~	~~Q2~~	Productivity	~~Enables users to more successfully share their media from mobile. Increases adoption of our ecosystem.~~	• ~~When attaching media or files, offer the opportunity to make use of Thunderbird Send~~ • ~~Consider upsell to paid tiers for users heavily using the feature.~~	~~Integrate with at least one of Thunderbird's Pro service offerings~~
Quick Filters and Email Actions	P2 High risk	Q3	Productivity	Focus on productivity by being able to quickly filter email using fixed criteria	• Subset of the quick filter actions you'd also see on desktop • Bring more common actions in closer reach to the users's fingers.	Introduce at least two new mechanisms that allow a user to focus on specific aspects of their email
Crash Reporting, Error Logging and Telemetry	P3	Q2	Foundations	As our users and community grows, we need to have easier mechanisms to identify faults and share them in a way they are actionable	• Select a suitable crash reporting service and integrate it into the app. • Implement the telemetry design and onboard users • Make it easier for users to share	Foundational Work

					logging with developers	
iOS Sending Email	P1	Q3	📋Productivity	Foundations for iOS	• Composing and sending email	Release a testing version of Thunderbird for iOS that allows you to read and send email.
~~Smart Email Summaries Experiment~~	~~P2~~	~~Q3~~	~~📋Productivity~~	~~Provide users with a better glance on what is in their inbox.~~	• ~~Make use of local AI on top devices to summarize email and provide an alternative email preview. Make sure not to market this as an AI feature~~ • ~~This feature can be kept on beta for a longer period of time to allow for more experimentation.~~ • ~~TODO smart subject instead? General time for experimentation~~	~~Successfully launch an experiment that makes use of AI to increase email productivity~~
Investigate integration Appointment	P3	Q3	📋Productivity	Great niche feature to immediately schedule events when meeting in person.	• Investigate if this is a good fit for Thunderbird Mobile. • ~~Prototype design~~	Prototype Experiments
HTML Signatures	P1	Q4	🎨 Window Paint	Highly requested on Mozilla Connect.	• Allow users to input signatures as HTML • First version to allow entering HTML tags • Second version (P3) to allow HTML compose	???
~~Send Later / Undo Send~~	~~P1~~ ~~High risk~~	~~Q4~~	~~📋Productivity~~	~~Highly requested on Mozilla Connect.~~	• ~~Implement an optional send delay where users can undo send~~ • ~~Make undo send available as an immediate action after sending an email.~~ • ~~Enable this feature by default with a low delay (e.g. 10 seconds)~~	~~Introduce at least two new mechanisms that allow a user to focus on specific aspects of their email.~~
iOS Encrypted Email	P3	Q4	📋Productivity	This is important for Thunderbird's general direction. Ensure encryption is a cornerstone of our product palette	• Decide on S/MIME vs OpenPGP • Implement reading encrypted email	Release a testing version of Thunderbird for iOS that allows you to read and send email. Note this will very much bleed into 2026, as we are scheduling to begin in December.

Additionally, we're looking to engage in the following refactoring projects that ease the implementation of the above. This is to fulfill the goal: "Complete and document at least four key architectural refactorings aimed at simplifying code and reducing technical debt".

NOTE: There are currently more refactoring projects on this list than planned for. We're looking to see where it makes sense to reduce the scope while still being able to deliver the features we need.

Refactoring	Priority	Quarter	Why	What	Enables Goals
Message Synchronization	P2	Q2	We have a number of support issues due to unexpected behavior of the message backend. Synchronization operations are not always deterministic, and messages are not always downloaded from the server	• Design an offline-first synchronization backend with consistent operational ordering and conflict resolution • Optimize message download operations and cache information in a way that filtering for certain types of messages becomes more simple to implement. • Develop the backend in a way that risk of data loss between implementations is minimized.	• Enables: Quick Filters • Goals: Increases trust in messages received
List view windowing	P2	Q2	The more messages we download the more it becomes a performance hit. Implementing windowing will allow us to overcome previous limits with message download while maintaining performance.	• Develop views on messages based on the new message database that show only a portion of the messages and offer lazy loading scrolling • TODO describe further	• Requires: Message Synchronization • Enables: Quick Filters
~~Account & App settings change notifications~~	~~P3~~	~~Q2~~	~~Account and application settings are not always applied immediately, because the code to notify for changes is not consistent.~~	• ~~Create an event-driven architecture or observer pattern to handle change notifications.~~ • ~~Change existing settings code to implement this new architecture.~~ • ~~Split settings into feature specific implementations where appropriate~~ • ~~Refactor in a way to reduce direct dependency on legacy code, specifically the K9 class.~~ • ~~Move away from mutable settings to immutability~~	• ~~Enables: Account Drawer improvements~~ • ~~Enables: Quick Filters~~ • ~~Enables: Lots of smaller community bugfixes to provide more UX consistency~~
~~Message Controller Refactor~~	~~P2~~	~~Q?~~	~~The message controller is a very central piece of infrastructure and used to show various error states. It is still written in Java and very tightly coupled.~~	• ~~Convert Java code to Kotlin~~ • ~~Split code into multiple files~~ • ~~Increase test coverage~~ • ~~Refactor in a way to reduce dependencies on legacy code~~	• ~~Enables: Notifications Overhaul~~
~~Convert screens from XML to Compose~~	~~P3~~	~~Q?~~	~~We're moving away from XML designs into Jetpack Compose. This won't be a simple task, but something repeatable we can do per use-case. It helps us better implement edge to edge support.~~	• ~~Determine a "model" conversion, a change from XML to Jetpack Compose that is predictable, not too complicated.~~ • ~~Document process of conversion — what to look out for — so this can become a task that contributors can take on.~~ • ~~Convert a select number of screens from XML to Jetpack Compose, guided by the screens that need work for edge to edge support.~~ • ~~Document/screenshot all the screens we have in the app. Maybe even automate the screenshot generation.~~	• ~~Enables: Android 15: Edge to Edge Support~~

				~~• Out of scope: Redesign all the screens.~~	
Message Compose	P3	Q3	The HTML parsing for sending emails needs improvement. We need better and more consistent ways to hook into the sending process.	• Make use of JSoup for HTML parsing across the board • Overhaul the way the HTML of the original message is modified to create the new message when replying • Overhaul the way the HTML of the original message is modified to create the new message when forwarding	• Enables: HTML Signatures
~~Retry Send Architecture~~	~~P2~~	~~Q3~~	~~Being able to retry sending email is very arbitrary at the moment, it is tied to fetching email. If we want to implement a reliable send later / undo send feature, we're going to need to separate this.~~	~~• Use WorkManager to send messages (including retries)~~ ~~• Investigate whether we should use WorkManager's "expedited work" mechanism~~ ~~• Add a mechanism to notify the user of the current state?~~	~~• Enables: Undo Send / Send Later~~

🏗️ Methodologies

This is how we'd like to get things done in the team. It gives each individual the autonomy to plan and execute their projects, while providing manager support and enabling cross- and intra-team collaboration efficiently. It allows us to focus on what is important and get the feedback we need from users.

1️⃣ You own the plan, from start to finish

- We are a small organization, which means you may have broader responsibilities than in other large companies where each function is separate.
- Make sure you look left and right of what you are doing to see if someone else is responsible for it. Do what needs to be done so you don't leave a gap:
    - Take it on yourself
    - Determine someone who should be responsible and reach out to them to make your case
    - Work with your manager to see who can take it on.
- You own status updates and discussions with the team. Don't just tell your manager what you did, keep the team informed in the operations meeting on what is next. Set up individual and group meetings as needed.
- If you need to delegate to others in the team or organization to get your goal done, you are responsible for delegating successfully.
    - Ensure delegated tasks are on track by maintaining open communication and providing support as needed.
    - If you are on the receiving end, think strategically: If you're saying yes to this, what are you saying no to?
- Your manager is your coach, not your problem solver. Use them to gain perspective, refine your approach and tackle the challenges in front of you. Your manager will help you see how what you are doing fits in the bigger picture.
- You can work with program management directly to map out the timeline and get other teams on board, either through the operations meeting or 1:1.

2 The time is now

- Assess whether a task is a priority or belongs in the 'someday' bucket.
    - If it's a priority, take initiative and integrate it into team/roadmap planning. See also 1
    - If not, how does talking about it impact our ability to focus and move forward?
- Strike a balance between getting every detail right and making progress in the bigger picture—neither can be fully optimized.

3 Always behind a feature flag

- New features or big rebuilds should always be behind a feature flag, to minimize risk and enable controlled user feedback.
- Feature flags can be exposed to the users or hidden, depending on the scenario.
- Keep the old code untouched until the new thing works, as much as possible

4 It is done when you ship it

- Plan before coding is mandatory. Document clear objectives and map out required tasks from other teams (e.g. in a Scope of Work)
- We develop incrementally. Ship your changes in smaller chunks so they can be tested early.
- Ship high-quality code that meets our definition of done, tested under feature flags in beta.
- You are responsible for the feedback cycle of getting it into production and incrementally improving documentation.

5 Daily Work vs Strategic Goals

- There are a number of tasks that belong to the day to day. We acknowledge they take time as well.
- A rough guideline is to reserve 60–70% of your week for work on goals, and 30–40% on day to day work.
- The following tasks are typical day to day work items.
    - Improving documentation
    - Fixing high impact or time sensitive bugs reported (though prefer adding them to the next sprint planning instead if severity allows)
    - Architectural support and mentoring for community members in the bug tracker
    - Troubleshooting and support as part of issue triage
- The following considerations are valuable while working on your goals
    - Legacy refactoring is part of the task and needs to be part of the timeline. If additional refactoring is advisable check with your manager on extending the schedule.
        - Start with planning what the ideal architecture looks like. Perfect is the enemy of good, but we should know where we are heading before we begin.
        - Reduce dependencies on legacy code and refactor into feature modules
        - Convert Java code to Kotlin when touching it.
    - When modifying code, make an effort to improve accessibility by following Mozilla's accessibility guidelines and the relevant mobile accessibility resources. While we are not aiming for full optimization for every possible accessibility scenario, we do expect the app to be functional and not frustrating for users with common accessibility needs.