Thunderbird Desktop - 2025 Product Roadmap

👁️Vision statement

⭐North Star

- Thunderbird is an extremely flexible communication and productivity tool.

🗺️Destination

- Thunderbird is a fast and multiplatform **email and calendar application**.
- **Email** usage is enhanced by vital features such as:
  - Thunderbird Send
  - Thunderbird Sync
- **Calendar** usage is enhanced by vital features such as:
  - Clean and pleasant UI
  - Thunderbird Appointment
  - Video conferencing integration
- Across all of it, Thunderbird offers a highly flexible and **customizable UI**, with sensible defaults, and easy onboarding.

👫Target Customers

✅DOs	❌DON'Ts
Become a strong cult that attracts multiple niches and user groups.	Go for the general population, trying to convince "normal" users to stop using webmail.

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

- Tech/Open Source enthusiasts
- Tech/Open Source professionals
- Privacy focused users
- Small and medium businesses
- Educational Institutions
- Government Agencies/Public sector

🌄Customers Horizons

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

- Retain users with convenience, honesty, and kindness.
- Do less (fewer projects per year), but do it better than everyone else (bring them to completion and make them stable and reliable).
- Keep the focus, accept new ideas but save them in a bucket for later prioritization.

📦Product Goals

✅DOs	❌DON'Ts
Be a first class citizen on every platform by improving **native OS integration**.	Become a "generalist" app that tries to do everything.
Ship **high quality UI and UX** changes.	Add hidden preferences for edge cases.
Make **customization** more powerful and easier to use/discover.	Data-loss! Seriously, stop!
Surpass our competitors in terms of **stability and convenience**.	
Powerful **API** for extensibility.	

Engineering focus

- Front-end features help us **market the product** better.
- Approaching every feature and initiative by focusing on the value that users can see with a strong focus on **privacy and user's control**.
- Removing technical debt is key to ensure the sustainability of the application, but core components rework has to be done through **careful and small iterations**, ensuring that the old system remains in place for easy rollbacks.
- Including the "extensibility" aspect early in the planning process is key to ensure **faster add-ons API implementation** and building community expectations.

Target Areas

We could list a million and more features that we would like to work on during this cycle, but we can only work on a limited amount, therefore we need to keep the focus on those target areas that will help move Thunderbird forward.

Stability

> Ensuring a stable, reliable, and predictable release it's vital for the user perception and continuous adoption of Thunderbird. Our old code doesn't guarantee any of that, and a large consistent effort needs to be directed to this initiative.

Encryption

Making encryption easily usable even to non technical users. Ensuring that Thunderbird champions this aspect of email security and introduces innovative solutions to drive adoption and retention.

Competition

Thunderbird needs to stop falling behind our competitors due to lack of simple features. We need to reach feature parity, whenever possible, with competitors for features that make sense and can offer a "familiar" experience to new users.

Innovation

We can't only catch up with the competition and refactor code. We need to also dedicate some time on innovating and exploring new solutions (outside and inside Thunderbird) to elevate our offering to the users and truly become an essential productivity tool.

Prioritization overview

The following **Prioritization** is meant to encapsulate 2 adjacent meanings:

- Importance of a feature for the success of Thunderbird
- Prioritization of allocation of resources

Prioritization doesn't strictly mean "*doing things in this order*", as the **Delivery Quarter** column doesn't always match the prioritization level.

Some of the features listed are small and easily achievable, which means some parallel efforts can always happen and multiple projects can be driven at the same time. The *P1s* are those efforts that, in case of major issues, roadblocks, lack of resources, or any other problem that might arise, will always remain the top focus while everything else can be postponed or cancelled if needed.

All of the efforts and features that are listed here are meant as an indication of direction and destination for the next year of Thunderbird.

If by the end of next year we only achieve the P1s in these lists, that's considered a success!

🎨Front-end priorities

Feature	Priority	Delivery Quarter	Target Areas	Why	What (high level objectives, not a complete list)
Conversation View [SoW] Conversation View v0.1	P1	Q1 | 2025	Competition	An email client without a conversation view in 2025 is just unacceptable.	• Full thread conversation in message pane. • Collapsible/Expandable messages. • Thread tree visualization. • Reply inline (optional).
Calendar UI/UX [SoW] Calendar UI	P1	Q2 | 2025	Stability Innovation Competition	Calendar pairs with email naturally, it needs to be a seamless and pleasant experience, and offer the same features of our competitors, if not better.	• Modern event modals with flexible UI. • Video conferencing integration. • Maps searching integration (Google, Apple, OpenStreetMap). • Modern HTML, markdown, plaintext editor. • Customizable UI with sensible defaults. • Customizable agenda and reminders. • Better handling of unprocessed invitations.
First time user experience - Create brief	P1	Q2 | 2025	Innovation Encryption	A bad first impression takes years to shake off.	• Account Hub for email, calendar, address book, feeds, and newsgroups. • Thunderbird Sync as the first option. • Guided discoveries of features when first accessing (welcome wizard). • Early customization options to set the desired defaults. • Expose important entry points like encryption and other more hidden settings.
Unified Toolbar completion - Create brief	P2	Q3 | 2025	Innovation	Let the rest of the UI consume the new Unified Toolbar offering a centralized place to customize all views.	• Remove old customizable UI toolbar. • Unifying controls for spaces, standalone dialogs, and windows. • Add overflow space. • Add more buttons and elements (spin buttons, separators, button groups, etc). • Menu position. • Rows. • Drop the App Menu!
Tabs - Create brief	P2	Q3 | 2025	Innovation	Let's rethink how tabs are implemented in Thunderbird to provide better workflows and extensibility through add-ons.	• Spaces and tabs rebuild, with each space handling their own tabs. • Standalone windows per space without the need of having the "mail" space.
OS System Tray SoW - Linux System Tray	P2	Q4 | 2025	Competition	Basic native integration that shows how behind we are compared to everyone else.	• Same tray code for Linux, Windows, and macOS. • Unread indicator and counter. • Minimize to tray. • Start minimized. • Compose action per account. • Open settings. • Quit action.
Native OS Notifications	P2	Q4 | 2025	Competition	Another basic native integration that	• Quick actions to reply, delete, mark messages.

Feature	Priority	Delivery Quarter	Target Areas	Why	What (high level objectives, not a complete list)
- Create brief				shows how behind we are compared to everyone else.	• Pause notifications for # time. • Manage notifications per account. • Focus modes.
Compose window - Create brief	P2	Q1 | 2026	Competition Stability	Writing an email should be easy, intuitive, and pleasant.	• Drop the C++ editor code. • Implement a modern JS editor component that can be reused in calendar, tasks, and any other textarea field that needs it (signature, address book notes, etc). • Link previews. • Rich text with advanced features. • Markdown support and preview. • Native emojis support (unicode). • Mentions support.
Settings UI - Create brief	P3	Q1 | 2026	Innovation	Requiring a textfield to search for settings is a UX failure.	• Modal dialog, because a settings tab doesn't make sense. • More sections and tabs for an intuitive discovery path, drop the never ending single pages. • All "Appearance" settings in one page, including native themes (follow system, light, dark) • Application settings and Account settings should be in the same place. • "Labs" section for listing experiments.
Tasks - Create brief	P3	Q2 | 2026	Competition	ToDo lists UX pairs naturally with email and calendar.	• Easier onboarding! Store them in the profile and sync them if there are no supported calendars. • Use open specs to allow interoperability. • Be opinionated with what we do here. • Modern UI. • Show tasks in the calendar. • Kanban board. • Integrate into emails, address books, and other areas.
Home space - Create brief	P3	Q3 | 2026	Innovation Competition	A better version of the account central for our start page and other useful sections.	• Stats! • Quick links to common actions. • Summarization of accounts. • Blog listing (feed). • Suggested actions.
Chat design kickoff - Create brief	P3	Q4 | 2025	Competition Encryption	Chat will be a core component of our Enterprise offering. Ending 2025 with a solid action plan and direction for 2026 will help us allocate resources and predict what we can offer to potential enterprise users.	• Create design plan • Identify areas with technical debt • Prioritize existing features • Prioritize new features

⚙️Back-end priorities

Feature	Priority	Delivery Quarter	Target Areas	Why	What (high level objectives, not a complete list)
Exchange Email Exchange	P1	Q1 | 2025	Competi...	Microsoft has +100M users locked into Exchange and their software is not great.	• Full support of email operations. • Full support of account creation.
Global Database - Create brief Mail back end long term plan Global Message Database	P1	Q2 | 2025	Stability	Our current back-end is just holding us back.	• SQLite based. • Rust interfaces. • Migrations. • Rollbacks. • Build in parallel and populate it alongside the current database.
A New Email Encryption - Create brief	P1	Q1 | 2026	Encrypti... Competi... Innovati...	Current email encryption solutions are cumbersome, requiring will and training time to use it. Thunderbird can be the driver and the champion of a new open implementation.	• Thunderbird should be the champion of encryption and propose a new open path. • Making it easier for non technical users to use a reliable encrypted solution.
Sentry - Create brief	P2	Q1 | 2025	Stability	Improving our awareness of errors across the application to more quickly react to issues that users might not see nor report.	• Add Sentry SDK & Initialise with DSN • Add channel/environment • Leverage default error capturing mechanisms • Add Custom Breadcrumb for one component • Configure reporting • Test • Document for future consumption in other components
Performance - Create brief	P2	Q2 | 2025	Stability	Optimize and rebuild to offer faster performance.	• Audit current startup code. • Slowly rebuild it. • Create performance profiles to measure startup performance impacts. • Implement the "start minimized" back-end. • Take ownership of the "primary password" process.
Exchange Calendar	P2	Q3 | 2025	Competi...	Fully supporting the Exchange protocol.	• Integrate the exchange calendar protocol. • Use it as a way to rebuild/improve the current calendar protocol. • Support exchange-specific features.
Exchange Address Book	P2	Q3 | 2025	Competi...	Fully supporting the Exchange protocol.	• Integrate the exchange address book protocol (maybe it uses carddav?) • Support exchange-specific features. • Fetch users' profile pictures.
Exchange Graph API - Create brief	P3	Q1 | 2026	Stability	Future proofing Exchange.	• Audit what can be supported via GraphAPI. • Build a parallel protocol to A/B test against EWS.
Notes	P3	Q2 | 2026	Innovati...	Increasing the productivity aspect of	• Store notes in the profile and in Sync.

Feature	Priority	Delivery Quarter	Target Areas	Why	What (high level objectives, not a complete list)
- Create brief				Thunderbird.	• HTML, markdown, plaintext editor. • Share notes via Send. • Copy the Google approach (create notes for this event). • Potential Pro offering (cloud notes with shared access like etherpad). • Potential syncing through IMAP.

⛔The obvious outsiders

As we focus on the P1s, we need to always account for the dozens of other features we currently ship in Thunderbird.

These features always require **maintenance, regressions and security fixes, and upstream porting**.

What we won't do during this cycle is adding new features or rebuilding big chunks of code outside of the prioritized components and efforts listed in the front-end and back-end tables.

These "outsiders" listed here are a way to highlight low priority maintenance efforts that can happen during the year, but that won't (shouldn't) take us away from the top priorities.

- **Feeds**
  - Inheriting the email UI for now is acceptable.
  - It needs a complete UI and UX overhaul.
- **Newsgroups**
  - Low user adoption.
  - Inheriting the email UI for now is acceptable.
- **Add-ons page**
  - A very large effort that will happen in the future.
  - It's tied to the website rebuild.
- **Address Book space**
  - More improvements are needed but the current state is superior to other areas.
  - A substantial rebuild happened 2 years ago.
- **OpenPGP and S/MIME integration**
  - Currently in a usable and enough stable state.
  - Dedicating security fixes during this cycle.
  - The UI saw some improvements in the past few years.
- **Message Filters**
  - They will need to be rebuilt from scratch.
  - Tied to the new database implementation for easier message move/copy handling.
- **Import/Export**
  - Sync will make the need to import/export the whole profile hopefully uncommon.
  - After that we should disable exporting the entire profile because it fails when dealing with large sizes.
- **Account Settings**
  - They should be integrated into the Settings space.

    - It needs a lot of UI/UX rethinking.

🏗️ Methodologies

Enforce a much stricter and predictable methodology and workflows. Freedom to work on random issues, and decide how to approach them independently should be granted by the managers, and the approach should be agreed upon between managers and tech leads.

1️⃣ Build things in parallel, don't replace core components

- Touching core code is a big red flag.
- Always build changes in parallel.
- Large clean ups should be first approved by the manager and the approach should be documented.
- Depending on the implementation, a rollback migration should be added as a fail safe.

2️⃣ Plan before code

- Align on coding approaches before working on patches.
- Align on architecture decisions and best practices.
- Share WIP patches to have quick sync ups before spending too much time on it.
- Write quick documentation of general approaches and work briefs to seek alignment.
- Add tests for code before removing/modifying it to ensure the same behavior is maintained during refactoring/clean-up.

3️⃣ Build our own debug tools (when needed)

- Our current architecture and storage is non-standard and hard to debug.
- We should build our own debug tools to remove barriers and frictions.
- Share our debug tools with our community.

4️⃣ Always behind a preference

- New features or big rebuilds should always be behind a preference.
- Preferences can be exposed to the users or hidden, depending on the scenario.
- Keep the old code untouched.

5️⃣ List new features inside a Labs settings section

- New features that need to be exposed to users should be listed inside the Settings as "experimental".
- New features should be toggleable.

6 Enable features programmatically in the various channels

- New features, even incomplete ones, should almost always be ON by default on Daily.
- Depending on the state and completion of a feature, it should be enabled on Beta.
- Enable on ESR by default only after the feature has been completed and passed through QA.

7 Post ESR clean up

- 3 months grace period to evaluate new features against our largest population.
- Plan for removal of the old feature.
- Remove the preference and make the new feature the only available default.

👨‍🔧 Maintenance

Feature	Priority	Quarter	Why	What (high level objectives, not a complete list)
Support URLs Support URLs in Thund...	P1	Q1 | 2025	Let's be consistent and not lead users to broken links.	• Plan and implement an efficient and standard approach to links within the application
Better logging/diagnostics -	P1	Q1 | 2025	We often find it hard to capture helpful information from users when issues are reported. To save time and frustration on both sides, it would help if our logging and diagnostics were improved.	• Implement Sentry and telemetry to capture data that will help us prioritize response to issues based on frequency. • Add or change logging tools to make it easier for the majority of users to provide logs.
Fluent migration Fluent Migration - Scop... Fluent Migration - Guid...	P2	Q2 | 2025	DTD and properties files will one day be removed.	• Assign a small migration effort every 2 weeks to each developer.
Complete Glean testing and uplift - Create brief	P2	Q2 | 2025	Fully commit to our new telemetry SDK	• Complete the project. • Validation that data is being captured as expected. • Help services consume the data.
Hardening	P2	Q3 | 2025		•
Fuzzing and Monkey testing - Create brief	P2	Q3 | 2025	Our current testing and QA process doesn't catch all issues and our users suffer.	• Work with Rob to create scripts, VMs, checking tools etc that we can have running regularly to catch edge cases much sooner without users having to experience then report.
Remove IMAP JS - Create brief	P3	Q3 | 2025	Old approach of rewriting IMAP in JS. Discarded by the core developers as not a viable approach.	• Remove unused code. • Remove preference. • Close all related open bugzilla bugs.

Feature	Priority	Quarter	Why	What (high level objectives, not a complete list)
Re-implement Movemail	P3	Q3 | 2025	We had a few thousands sysadmins using it, now they all moved to Betterbird. That's just a silly loss for a feature that had no maintenance cost.	• Reimplement the ability to use Movemail in Thunderbird. • No account creation feature for now.
Removal of the *Overlay.js files - Create brief	P3	Q4 | 2025	Overlay files come from a time when modules weren't a thing.	• Replace overlay files with modules and system modules. • Defer non-DOM operations to system modules. • Lazy load modules when needed.