TIEASY

Cursor Master Build Specification — Version 1.0

Mobile-first task management web application | CodeIgniter 4 + PHP + MySQL

Purpose: This is the functional and technical authority for Cursor. The supplied wireframe is the visual/UI authority.

1. Build Authority and Non-Negotiable Rules

- Project name is TIEASY.
- The supplied wireframe is already finalized. Cursor must not redesign it.
- The wireframe controls layout, visual style, spacing, colors, typography, icons and screen structure.
- This specification controls functionality, validation, database behavior, authentication, recurrence, filters, search and backend logic.
- If the wireframe lacks a functional element explicitly required here, add it with the least possible visual disturbance.
- The app is web-based but must always retain a mobile-app-style presentation, including on desktop.
- Do not introduce React, Vue or Angular. Use CodeIgniter 4, PHP, MySQL, Bootstrap 5 where useful, vanilla JavaScript/AJAX and Chart.js.
- Do not leave actionable controls as visual-only placeholders.

2. Recommended Technology Stack

Layer	Technology
Backend	CodeIgniter 4, PHP 8.2+
Database	MySQL 8+
Frontend	HTML/CSS, Bootstrap 5 where useful, vanilla JavaScript/AJAX
Charts	Chart.js
OTP Email	CI4 Email / SMTP
Authentication	CI4 session + secure persistent login token
Scheduled jobs	CodeIgniter Spark CLI commands + cron

3. Application Entry, Sign In and OTP

3.1 Default Entry Screen

The current wireframe opens the Dashboard directly. This must be corrected. When there is no authenticated/persistent login, TIEASY must open the Sign In screen by default. Protected URLs must redirect unauthenticated users to Sign In.

3.2 Email Instead of Mobile Number

The existing Sign In wireframe was designed for mobile number + OTP. Keep the same visual design but replace the mobile-number field with Email Address. The existing OTP screen may remain visually the same, but all SMS/mobile wording must be changed to email wording.

3.3 OTP Flow

1. User enters a valid email address.
2. Generate a random 6-digit OTP.
3. Store only a secure OTP hash where practical.
4. Send OTP to the entered email using configured SMTP.
5. OTP expires after 10 minutes.

6. Maximum 5 verification attempts per OTP.
7. On success, invalidate the OTP and identify/create the user.
8. A new user goes to Profile onboarding. An existing onboarded user goes to Dashboard.

3.4 Resend OTP

- OTP screen must contain Resend OTP.
- Use a 60-second cooldown and show a countdown such as 'Resend OTP in 00:42'.
- After cooldown, enable Resend OTP.
- Resending generates a new OTP and invalidates the previous active OTP.
- Restart expiry and resend cooldown.
- Enforce cooldown/rate limiting on the backend, not only JavaScript.
- Recommended protection: maximum about 5 OTP sends in a rolling 30-minute period.

3.5 OTP Errors

Condition	User-facing message
Incorrect OTP	Incorrect OTP. Please try again.
Expired OTP	This OTP has expired. Please request a new OTP.
Too many attempts	Too many incorrect attempts. Please resend OTP.
Resend too soon	Please wait before requesting another OTP.
Email send failure	We could not send the OTP. Please try again.

4. Persistent Login and Logout

Business requirement: once logged in, the user should remain signed in until the user explicitly logs out, including after closing and reopening the browser.

- Use a normal CI4 session for the active browser session plus a secure long-lived persistent-login token in an HttpOnly cookie.
- Do not rely on an endlessly reusable PHP session ID.
- Use Secure cookies under HTTPS, SameSite protection, token hashing and session regeneration.
- Do not store raw reusable authentication secrets in localStorage.
- If the CI4 session is gone but a valid persistent token exists, validate it, restore the authenticated session and open Dashboard.
- Logout must destroy the CI4 session, revoke/remove the persistent token and cookie, and redirect to Sign In.
- Browser Back after logout must not reveal protected content; server-side auth checks remain mandatory.

4.1 More/Hamburger Menu

The authenticated hamburger menu must contain: Notifications, Category Manager, Settings, Contact Support, Logout. Replace any authenticated-state 'Sign In' item with Logout.

5. First Login Profile Onboarding

After successful OTP verification, a new user must see the Profile screen before Dashboard. All profile fields are optional and the existing Skip link must remain. Save or Skip both complete onboarding and then open Dashboard. Returning users must not be forced through onboarding again.

Field	Type	Required	Rules
Profile Name	Text	No	Trim; max 100 chars; allow normal name punctuation; reject whitespace-only value.
Age Group	Dropdown	No	Under 15; 15–24; 25–34; 35–44; 45–54; 55–64; 65+.
Gender	Dropdown	No	Male; Female; Non-binary;

			Prefer not to say; Other.
Occupation	Text	No	Trim; max 100 chars; allow letters, numbers and normal punctuation.
Phone	Telephone/Text	No	New field; no OTP verification in V1; allow international-style numbers; max about 20 chars.
Email	Read-only	No	Autofill from verified login email; never editable here.

Store profile_onboarding_completed = 1 when the user either saves the onboarding screen or clicks Skip. Settings/Profile may later edit Profile Name, Age Group, Gender, Occupation and Phone; verified Email remains read-only.

6. Dashboard / Home

After authentication/onboarding, Dashboard is the main screen. Preserve the supplied wireframe.

- Top task search field.
- Three filters: Date, Status and Category.
- Applied-filter text/labels below the search/filter controls.
- Four priority cards in a 2 × 2 mobile layout.
- Default filter state: Today + Pending + All Categories.
- Bottom navigation must remain exactly four actions as defined later.

7. Priority Cards and Task Display

Priority ID	Priority
1	Important & Urgent
2	Important, Not Urgent
3	Not Important, Urgent
4	Not Important, Not Urgent

The existing 'Suggested by AI' text is of no use and must be completely removed. Do not add task counts if they are not present in the wireframe.

7.1 Normal Dashboard Card / Normal Priority Detail

```
Category Name • Due Date • Due Time
Task Title
```

Only show date/time pieces that exist. Examples: 'Work • 29 Aug • 4:30 PM', 'Work • 29 Aug', or simply 'Work'. Never render empty separators. Priority is not repeated inside a normal priority-specific view.

Each priority card retains its existing Add Task action. When Add Task is opened from a particular priority card, preselect that priority where practical.

8. Filters

8.1 Date Filter

- Today
- Tomorrow
- This Week
- Next 7 Days
- This Month

- Custom Date (From Date + To Date)

Due Date is optional. A task without a due date must not be artificially assigned a date. Date-specific filters such as Today or Tomorrow naturally exclude undated tasks. Where the wireframe supports an all-inclusive date state, undated tasks may be included there.

8.2 Status Filter

- Pending
- Completed
- All

Default is Pending.

8.3 Category Filter

Options are All Categories plus categories created by the logged-in user. Default is All Categories. All three filters must work simultaneously.

9. Search Behavior — Cross-Priority

Search must not open a specific priority detail because matching tasks can belong to multiple priorities.

9. User enters search text and searches.
10. Open the same Task List/Card Detail style screen used by the wireframe.
11. Where a priority heading normally appears, show the user's search text instead.
12. Return matching tasks across all four priorities for the logged-in user.
13. Use case-insensitive partial matching on Task Title.
14. Search should respect relevant active filters unless the wireframe flow explicitly clears them.

9.1 Search Result Task Layout

```
Category Name • Due Date • Due Time
Task Title
Priority
```

Priority is intentionally shown in search results because results may span multiple priorities. It is intentionally omitted from normal priority detail.

10. Create Task

Field	Required	Rules
Task Title	Yes	Trim; max 255 chars.
Due Date	No	Nullable. Many tasks may have no due date.
Due Time	No	If there is no due date, do not store an isolated due time.
Mark as Recurring	No	Toggle; OFF by default.
Priority	Yes	One of the four defined priorities.
Category	Yes	Select from logged-in user's categories.

Create Task can be opened from the footer + icon or the Add Task action on each priority card.

11. Recurring Tasks

Use a parent task + recurrence rule + occurrence model. Do not create unrelated duplicate master tasks.

11.1 Daily

Support every N days: every day, every 2 days, every 3 days, etc.

11.2 Weekly

Support every N weeks plus one or more selected weekdays: Monday through Sunday. At least one weekday is required.

11.3 Monthly

- Day-of-month method: e.g. day 15 every month, day 25 every 2 months.
- Weekday-position method: First/Second/Third/Fourth/Last + Monday–Sunday, e.g. First Wednesday or Last Friday.

11.4 Recurring Time

Recurring task time is optional. If absent, generated occurrences are date-only.

11.5 Occurrence Generation

Generate recurring occurrences approximately 90 days ahead and extend them using a daily CLI/cron job. Avoid duplicates. Non-recurring tasks receive one occurrence.

11.6 Completion and Deletion

Completing one occurrence must not complete the recurring series. For recurring deletion, implement both:

- Delete this occurrence
- Delete this and future occurrences

The second option must preserve historical occurrences and stop/remove the selected and future schedule, for example by setting an end/stop date and removing future generated occurrences.

12. Swipe Actions

In priority/search task lists, preserve the mobile swipe behavior from the wireframe. Swiping reveals Complete and Delete actions.

- Complete: mark occurrence completed, store completed_at, remove immediately from Pending view, retain in Completed/All and analytics.
- Delete: non-recurring task asks for confirmation; recurring task offers the two recurrence deletion choices above.

13. Category Manager

- View categories
- Add category
- Edit category
- Delete category

Category Name is required and logically unique per user. Different users may use the same category name. All category access must be user-scoped.

14. Bottom Navigation — Exact Four Actions

#	**Footer Action**	**Behavior**
1	Home	Open Dashboard.
2	Analytics	Open Analytics.
3	Hamburger / Three Lines	Open Notifications, Category

		Manager, Settings, Contact Support, Logout.
4	+	Open Add Task.

Do not redesign this into five or more permanent footer icons. The wireframe already contains this structure.

15. Notifications

Version 1 uses in-app notifications only. Do not implement Firebase/FCM.

- Add a small bell/notification icon unobtrusively near the top-right of the authenticated app header because the current wireframe has no dedicated notification icon.
- Bell may show an unread badge/indicator.
- Notifications must also remain accessible from the hamburger menu.
- For due tasks, create appropriate in-app notifications using a scheduled CLI/cron job.
- Tasks without due dates do not create due notifications.
- Prevent duplicate notifications for the same occurrence.
- Track read/unread and read timestamp.

16. Analytics

- Completed vs Pending: count + percentage, pie/doughnut chart.
- Task by Priority: count + percentage for all four priorities.
- Task by Category: pending task count + percentage grouped by category.
- Deleted occurrences are excluded.
- Only the logged-in user's data is included.

17. Contact Support

Contact Support must contain Profile Name, Email, Phone No, How Can We Help?, Message and Submit.

Field	Prefill/Edit Rule	Submission Rule
Profile Name	If saved in profile: prefill + read-only. If missing: blank + editable.	Required for support submission.
Email	Always prefill verified login email + read-only.	Required.
Phone No	If saved in profile: prefill + read-only. If missing: editable.	Optional unless changed later.
How Can We Help?	Database-driven support category dropdown.	Required.
Message	Editable textarea.	Required.

Important: entering a missing Profile Name or Phone in Contact Support must NOT automatically update the permanent user profile. Store the submitted values as a snapshot on the support request.

17.1 Support Categories

Create database-driven support_categories. Initial seed examples: Technical Issue, Account/Login Issue, Task Management, Notification Issue, Feedback/Suggestion, Other.

18. Database Schema

18.1 users

```
id BIGINT UNSIGNED PK AI
email VARCHAR(190) NOT NULL UNIQUE
```

```
profile_name VARCHAR(100) NULL
age_group VARCHAR(30) NULL
gender VARCHAR(30) NULL
occupation VARCHAR(100) NULL
phone VARCHAR(30) NULL
profile_onboarding_completed TINYINT(1) NOT NULL DEFAULT 0
is_active TINYINT(1) NOT NULL DEFAULT 1
last_login_at DATETIME NULL
created_at DATETIME NULL
updated_at DATETIME NULL
```

18.2 email_otps

```
id BIGINT UNSIGNED PK AI
email VARCHAR(190) NOT NULL
otp_hash VARCHAR(255) NOT NULL
attempt_count INT NOT NULL DEFAULT 0
expires_at DATETIME NOT NULL
verified_at DATETIME NULL
created_at DATETIME NULL
```

18.3 user_login_tokens

```
id BIGINT UNSIGNED PK AI
user_id BIGINT UNSIGNED NOT NULL
selector VARCHAR(64) NOT NULL UNIQUE
token_hash VARCHAR(255) NOT NULL
expires_at DATETIME NULL
last_used_at DATETIME NULL
created_at DATETIME NULL
```

18.4 categories

```
id BIGINT UNSIGNED PK AI
user_id BIGINT UNSIGNED NOT NULL
name VARCHAR(100) NOT NULL
is_active TINYINT(1) NOT NULL DEFAULT 1
created_at DATETIME NULL
updated_at DATETIME NULL
deleted_at DATETIME NULL
UNIQUE logical key: user_id + name
```

18.5 tasks

```
id BIGINT UNSIGNED PK AI
user_id BIGINT UNSIGNED NOT NULL
category_id BIGINT UNSIGNED NOT NULL
title VARCHAR(255) NOT NULL
priority TINYINT NOT NULL
due_date DATE NULL
due_time TIME NULL
is_recurring TINYINT(1) NOT NULL DEFAULT 0
recurrence_ends_at DATE NULL
created_at DATETIME NULL
updated_at DATETIME NULL
deleted_at DATETIME NULL
```

18.6 task_recurrence_rules

```
id BIGINT UNSIGNED PK AI
task_id BIGINT UNSIGNED NOT NULL
recurrence_type ENUM('daily','weekly','monthly') NOT NULL
interval_value INT NOT NULL DEFAULT 1
weekly_days JSON NULL
monthly_type ENUM('day_of_month','weekday_position') NULL
```

```
monthly_day TINYINT NULL
monthly_week_position ENUM('first','second','third','fourth','last') NULL
monthly_weekday TINYINT NULL
recurrence_time TIME NULL
start_date DATE NOT NULL
end_date DATE NULL
is_active TINYINT(1) NOT NULL DEFAULT 1
created_at DATETIME NULL
updated_at DATETIME NULL
```

Weekday mapping: 1=Monday, 2=Tuesday, 3=Wednesday, 4=Thursday, 5=Friday, 6=Saturday, 7=Sunday.

18.7 task_occurrences

```
id BIGINT UNSIGNED PK AI
task_id BIGINT UNSIGNED NOT NULL
occurrence_date DATE NULL
occurrence_time TIME NULL
status ENUM('pending','completed') NOT NULL DEFAULT 'pending'
completed_at DATETIME NULL
is_deleted TINYINT(1) NOT NULL DEFAULT 0
created_at DATETIME NULL
updated_at DATETIME NULL
```

18.8 notifications

```
id BIGINT UNSIGNED PK AI
user_id BIGINT UNSIGNED NOT NULL
task_id BIGINT UNSIGNED NULL
occurrence_id BIGINT UNSIGNED NULL
title VARCHAR(190) NOT NULL
message TEXT NOT NULL
is_read TINYINT(1) NOT NULL DEFAULT 0
created_at DATETIME NULL
read_at DATETIME NULL
```

18.9 support_categories

```
id BIGINT UNSIGNED PK AI
title VARCHAR(150) NOT NULL
is_active TINYINT(1) NOT NULL DEFAULT 1
sort_order INT NOT NULL DEFAULT 0
created_at DATETIME NULL
updated_at DATETIME NULL
```

18.10 support_requests

```
id BIGINT UNSIGNED PK AI
user_id BIGINT UNSIGNED NOT NULL
profile_name VARCHAR(150) NOT NULL
email VARCHAR(190) NOT NULL
phone VARCHAR(30) NULL
support_category_id BIGINT UNSIGNED NOT NULL
message TEXT NOT NULL
status ENUM('open','in_progress','resolved','closed') NOT NULL DEFAULT 'open'
created_at DATETIME NULL
updated_at DATETIME NULL
```

19. CI4 Application Structure

```
app/
  Controllers/
    AuthController.php
    DashboardController.php
```

```
    TaskController.php
    CategoryController.php
    AnalyticsController.php
    NotificationController.php
    SupportController.php
    SettingsController.php
  Models/
    UserModel.php
    EmailOtpModel.php
    UserLoginTokenModel.php
    CategoryModel.php
    TaskModel.php
    TaskRecurrenceModel.php
    TaskOccurrenceModel.php
    NotificationModel.php
    SupportCategoryModel.php
    SupportRequestModel.php
  Services/
    OtpService.php
    PersistentLoginService.php
    RecurrenceService.php
    NotificationService.php
  Filters/
    AuthFilter.php
  Views/
    auth/
    profile/
    dashboard/
    tasks/
    categories/
    analytics/
    notifications/
    support/
    settings/
    layouts/
  Database/
    Migrations/
    Seeds/
  Commands/
    GenerateTaskOccurrences.php
    GenerateTaskNotifications.php
```

20. Routes and Access

Public routes: Sign In, Send OTP, OTP Verification, Resend OTP. Everything else is authenticated.

```
/sign-in
/auth/send-otp
/auth/verify-otp
/auth/resend-otp

/profile/onboarding
/dashboard
/tasks/*
/categories/*
/analytics
/notifications/*
/support/*
/settings/*
/logout
```

21. Scheduled Commands

```
php spark tasks:generate-occurrences
php spark tasks:notifications
```

The occurrence command keeps active recurring tasks generated approximately 90 days ahead. The notification command creates due in-app notifications and prevents duplicates.

22. Security and Data Isolation

- Every task/category/notification/analytics query must be scoped to the authenticated user.
- Never trust a browser-supplied task/category/occurrence ID without verifying ownership.
- Enable CSRF protection.
- Validate all form inputs server-side even when JavaScript validation exists.
- Escape output as appropriate.
- Regenerate session ID after authentication.
- Hash OTPs/tokens where practical and never expose SMTP/server errors to users.
- Use prepared/query-builder operations rather than concatenating untrusted SQL.

23. AJAX Behavior

Use AJAX where practical for dashboard filtering, task search, completion, deletion and marking notifications read. Every AJAX endpoint must still validate authentication, CSRF and record ownership.

24. Development Sequence for Cursor

Phase	Work
1. Foundation	CI4 project, .env example, DB config, migrations, models, shared mobile layout, AuthFilter.
2. Authentication	Email OTP, resend OTP, persistent login, logout.
3. Profile	First-login onboarding, Skip, Settings/Profile.
4. Categories	List/Add/Edit/Delete.
5. Tasks	Create/Edit, optional due date/time, priority, category.
6. Recurrence	Daily/weekly/monthly rules, occurrences, completion, both deletion modes.
7. Dashboard	Four cards, metadata, filters, remove Suggested by AI.
8. Search	Cross-priority search result screen and priority display.
9. Analytics	Three analytics sections with Chart.js.
10. Notifications	Bell, list, read/unread, due-notification cron.
11. Support/Settings	Support categories, support form, settings.
12. Audit	Wireframe-to-route audit and full functional acceptance audit.

25. Functional Acceptance Checklist

☐ 1. Unauthenticated app opens Sign In, not Dashboard.

☐ 2. Sign In asks for Email instead of Mobile Number.

☐ 3. OTP is sent by email and OTP screen contains no misleading SMS/mobile wording.

☐ 4. Resend OTP exists, respects cooldown, invalidates old OTP and is backend rate-limited.

☐ 5. New user goes to optional Profile onboarding; Save or Skip goes to Dashboard.

☐ 6. Returning onboarded user goes directly to Dashboard.

☐ 7. All Profile fields are optional; Email is verified/read-only; Phone is added without phone OTP.

☐ 8. Persistent login restores access after browser close/reopen until explicit Logout.

☐ 9. Hamburger menu contains Notifications, Category Manager, Settings, Contact Support and Logout.

☐ 10. Logout revokes session/persistent login and protected pages redirect to Sign In.

☐ 11. User can add/edit/delete own categories.

☐ 12. User can create a task with no due date.

☐ 13. User can create a task with date only or date + time.

☐ 14. All four priorities work.

☐ 15. Daily, every-N-days, weekly, every-N-weeks, multiple weekdays, monthly day-of-month and monthly weekday-position recurrence work.

☐ 16. Completing one recurring occurrence does not complete future occurrences.

☐ 17. Delete this occurrence works.

☐ 18. Delete this and future occurrences preserves history and removes/stops future occurrences.

☐ 19. Dashboard defaults to Today + Pending + All Categories.

☐ 20. Date, status and category filters work individually and together.

☐ 21. 'Suggested by AI' is completely removed.

☐ 22. Normal task display shows Category plus optional Due Date/Time, then Task Title.

☐ 23. No task count is added unless already present in the wireframe.

☐ 24. Search works across priorities using case-insensitive partial title matching.

☐ 25. Search-result heading shows search text instead of a priority.

☐ 26. Search results show Category/date/time, Task Title and Priority.

☐ 27. Normal priority detail does not repeat Priority per task.

☐ 28. Swipe Complete and Delete actions work.

☐ 29. Completed tasks disappear from Pending immediately and remain under Completed/All.

☐ 30. Analytics counts/percentages are correct and deleted occurrences are excluded.

☐ 31. Bell icon exists near top-right and in-app notifications can be read/marked read.

☐ 32. Footer remains exactly Home, Analytics, Hamburger, +.

☐ 33. Contact Support pre-fills saved Profile Name/Phone as read-only, leaves missing values editable, and always shows verified Email read-only.

☐ 34. Support categories are database-driven.

☐ 35. Support form values are stored as a snapshot and do not silently update the user's profile.

☐ 36. One user can never access another user's data.

☐ 37. App remains mobile-style on desktop.

☐ 38. Every clickable wireframe control is implemented or explicitly documented as deferred.

26. Final Instruction to Cursor

Build TIEASY according to this specification and the supplied wireframe link/files. The wireframe is the UI authority; this document is the functional and technical authority.

- Do not redesign the wireframe.
- First inspect every wireframe screen and create a screen/route/action inventory before coding.
- Then implement phase-by-phase in the order above.
- Maintain a checklist using [Completed], [Pending], [Blocked].
- Do not use dummy/static data after database integration.
- Do not skip backend validation because frontend validation exists.
- At the end, perform both a Wireframe-to-Implementation Audit and the Functional Acceptance Checklist.
- If the wireframe and this document appear to conflict, preserve the visual design but follow this document for the corrected functional behavior unless the conflict makes implementation impossible; in that case, flag it before changing the design.

27. Prompt to Paste into Cursor

```
I am attaching the TIEASY Cursor Master Build Specification and providing the finalized wireframe
link.

WIREframe: <PASTE YOUR WIREFRAME LINK HERE>

Treat the wireframe as the UI/visual authority and the Master Build Specification as the
functional/technical authority.

IMPORTANT:
1. Do not redesign the UI.
2. Before writing application code, inspect all wireframe screens and create a screen-to-route-to-
action inventory.
3. Compare that inventory against the Master Build Specification and list any genuine conflict or
missing wireframe state.
4. Then build the project in the exact development sequence defined in the specification.
5. Use CodeIgniter 4 + PHP + MySQL + vanilla JS/AJAX + Chart.js.
6. Keep the application mobile-style even on desktop.
7. Do not leave buttons/icons as visual-only controls.
8. After each phase, update a [Completed] [Pending] [Blocked] checklist.
9. At the end, run the complete wireframe-to-implementation audit and functional acceptance
checklist.
10. Do not make independent product/UI decisions when the wireframe or specification already
defines the behavior.

Start by analyzing the wireframe and specification. Do not start coding until you have produced
the screen/route/action inventory and identified any conflicts.
```

End of TIEASY Cursor Master Build Specification.