TimeSheet Management — Functional Specification

1. Domain Overview

Timesheet is the single most-used feature of ICEBERG, accounting for **51.8% of all production traffic** (200,632 out of 1,075,687 requests across 3 years of production data). It is the entry point for all billable and non-billable time that flows through the approval pipeline into ERP exports.

The domain encompasses:

- **Daily time entry** against project/task/activity combinations
- **Overtime recording** with separate internal/external billing coefficients
- **Monthly aggregation** with visual breakdown of regular and overtime hours
- **Unfilled date tracking** for compliance monitoring
- **Time lock enforcement** across three hierarchical levels

2. UI Inventory

Pages by Traffic Volume

Rank	Page	Requests	% of ASPX	File Size	Purpose
1	`TimeSheet3.as	152,161	14.1%	—	Weekly time en
2	`UserTime.aspx	37,470	3.5%	—	Exception + ov
3	`pages/TimeShe	33,708	3.1%	—	Alternate path
4	`ManageTime.as	7,243	0.7%	—	Manager time m
5	`TimesheetMont	6,299	0.6%	—	Monthly summar
6	`TimesheetUnfi	4,673	0.4%	—	Gap tracking
7	`TimeSheetByM	—	—	—	Monthly view w
8	`Overtime.aspx	—	—	—	Overtime entry
9	`ManageActivit	—	—	—	Activity defin
10	`ManageActivit	—	—	—	Activity set m

TimeSheet3.aspx — Weekly Grid Layout

```
┌─────────────────────────────────────────────────────────────────┐
│ [← Previous Week]  Week of 03 Feb 2025  [Next Week →]          │
├──────────┬──────┬──────────┬────┬────┬────┬────┬────┬────┬────┤
```

```
│ Project   │ Role │ Activity │ Mo │ Tu │ We │ Th │ Fr │ Sa │ Su │ Tot │
├───────────┼──────┼──────────┼────┼────┼────┼────┼────┼────┼────┼─────┤
│ [dropdown]│[dd]  │ [dd]     │ [8]│ [8]│ [8]│ [8]│ [8]│ [ ]│ [ ]│ 40  │
│ [dropdown]│[dd]  │ [dd]     │ [4]│ [4]│ [4]│ [4]│ [4]│ [ ]│ [ ]│ 20  │
├───────────┴──────┴──────────┼────┼────┼────┼────┼────┼────┼────┼─────┤
│                  Day Totals │ 12 │ 12 │ 12 │ 12 │ 12 │  0 │  0 │ 60  │
└─────────────────────────────┴────┴────┴────┴────┴────┴────┴────┴─────┘
[+ Add Row]                                             [Save] [Reset]
```

TimeSheetByMonth.aspx — Monthly View Layout

```
┌──────┬───────┬──────────┬──────────┬──────────────────────┬──────────────────
│ Date │ Hours │ Comment  │ Overtime │ Internal (Rate×Res)  │ External (Rate×Re
├──────┼───────┼──────────┼──────────┼──────────────────────┼──────────────────
│ 1    │ 8     │          │ 2        │ 1.5 × 3.0            │ 2.0 × 4.0
│ 2    │ 8     │          │ —        │ —                    │ —
│ ...  │       │          │          │                      │
│ 28*  │ 8     │          │ 3        │ 1.0 × 3.0            │ 1.0 × 3.0
├──────┼───────┼──────────┼──────────┼──────────────────────┼──────────────────
│ TOTAL│ 160   │          │ 20       │ 50                   │ 60
└──────┴───────┴──────────┴──────────┴──────────────────────┴──────────────────

* Days with multiple overtime entries show `*` indicator and are non-editable
```

UserTime.aspx — Exception & Overtime Management

```
┌──────────────────────────────────────────────────────┐
│ Calendar View                                        │
│                                                      │
│ ┌───┬───┬───┬───┬───┬───┬───┐                        │
│ │Mo │Tu │We │Th │Fr │Sa │Su │  Legend:               │
│ │   │   │ V │ V │   │   │   │  V = Vacation          │
│ │   │   │   │   │   │   │   │  S = Sick Leave        │
│ │ S │ S │   │   │   │   │   │  B = Business Trip     │
│ └───┴───┴───┴───┴───┴───┴───┘  H = Holiday           │
│                                                      │
│ Summary: Sick: 2d | Vacation: 2d | Unpaid: 0d        │
├──────────────────────────────────────────────────────┤
│ Exception List                                       │
│ Type       │ From    │ To      │ Project │ [Delete]  │
│ Vacation   │ 05 Feb  │ 06 Feb  │ —       │ []        │
│ Sick Leave │ 10 Feb  │ 11 Feb  │ —       │ []        │
├──────────────────────────────────────────────────────┤
│ Overtime Entries                                     │
│ Date    │ Hours │ k_int │ k_ext │ Comment │ [Delete] │
│ 03 Feb  │ 2.0   │ 1.5   │ 2.0   │ Sprint  │ []       │
└──────────────────────────────────────────────────────┘
```

3. Business Rules

BR-001: Internal Project Marker [GPT-5 + NotebookLM]

`INTERNAL_ID = -1`. Hours logged against task ID −1 are internal/non-billable. This is a sentinel value used throughout the codebase to distinguish internal time from project-billed time.

Source: `TimeSheetByMonthMgr.cs`, `TimeSheet3.aspx.cs`

BR-002: Regular Day Counting [GPT-5 + NotebookLM]

The `CountInRegular` method defines what counts as a "regular" working day:

- Holiday days with a `WorkOff` exception → count as regular (employee is working on a non-working day)
- **Vacation days do NOT count as regular** (even though the employee is "away", it's not a working day)

Source: `TimeSheetByMonthMgr.cs / CountInRegular`

BR-003: Multiple Overtime Lock [GPT-5 + NotebookLM]

When a day has multiple overtime entries (potentially at different rates), the day becomes non-editable (`DayAvailable = false`) and displays a * indicator. This prevents accidental modification of mixed-rate overtime.

Source: `TimeSheetByMonthMgr.cs`

BR-004: Approval Read-Only Lock [GPT-5 + NotebookLM]

If **either** a division-level OR project-level approval exists for a month period, the **entire month** becomes read-only for that user. This is implemented in the `isPeriodApproved` method.

Note: This is OR logic, not AND. Either approval type alone is sufficient to lock the period.

Source: `TimeSheetByMonthMgr.cs / isPeriodApproved`

BR-005: Week-Aligned Persistence [GPT-5 + NotebookLM]

When saving monthly timesheet data, the system splits the month into **week-aligned periods** (7-day blocks starting Monday) before persisting. Each week is saved independently.

Source: `TimeSheetByMonthMgr.cs / SaveTimeSheetData`

BR-006: Overtime Delete-and-Recreate [GPT-5 + NotebookLM]

Overtime records are persisted using a delete-and-recreate pattern, not an update-in-place. When overtime is saved:

1. All existing overtime records for the period are deleted
2. New records are inserted with current values
3. Default comment applied if none provided: `"Overtime has been added on month time sheet page."`

Source: `TimeSheetByMonthMgr.cs`

BR-007: Null Coefficient Handling [GPT-5 + NotebookLM]

Null overtime coefficients (`k_internal`, `k_external`) are treated as **zero**, not as 1.0. This means overtime with null coefficients generates no billable or internal time.

Source: `TimeSheetByMonthMgr.cs`

BR-008: Soft Delete Then Insert [GPT-5 + NotebookLM]

Timesheet save operation:

1. Marks all existing hours for the week as `IsDeleted = true`
2. Inserts new hour records for the current week
3. This runs per week within the month

This pattern avoids update conflicts but increases database write volume and requires careful handling of the `IsDeleted` flag in queries.

Source: `TimeSheet3.aspx.cs / SaveTimeSheet`

BR-009: Calendar Force-Set Hours [GPT-5]

When setting a date exception via the calendar:

- `DayType.Workday` → forces **8 hours** for that day
- `DayType.Holiday` → forces **0 hours** for that day

Source: `TimeMgr.cs / SetDateException`

BR-010: WorkOff Inversion [GPT-5]

A `WorkOff` exception causes a user to be considered "working" even on a holiday or weekend day. It inverts the default non-working status. The `IsReallyWorking` method returns `true` for days with WorkOff exceptions regardless of calendar status.

Source: `TimeMgr.cs / IsReallyWorking`

BR-011: Working Day Arithmetic [GPT-5]

All scheduling arithmetic (adding/subtracting days) operates on **working days only**, skipping non-working days according to the branch-specific calendar. The `AddDate` method iterates through calendar days, counting only those where the branch marks the day as working.

Source: `TimeMgr.cs / AddDate`

---

4. Validation Rules

VR-012: Overtime Value Constraints [GPT-5]

Three constraints on overtime:

- Overtime holiday worked time must be **positive** (> 0)
- Regular worked time cannot be **negative** (≥ 0)
- Budget coefficients must be **≥ 0**

Source: `Consts.cs`

VR-013: Future Date Prohibition [GPT-5]

Tasks cannot be:

- Set in the **future** (beyond current date)
- Outside the user's **working period** (employment start to deactivation date)

Source: `Consts.cs`

VR-014: Business Trip Prerequisites [GPT-5]

A business trip exception requires:

1. The user must have **active projects** at that time
2. A specific **project must be selected** for the trip

Source: `Consts.cs`

VR-015: 1C Import Protection [GPT-5]

Exceptions imported from the 1C ERP system (identified by having a populated `iddoc` field) **cannot be deleted** unless explicitly overridden with `checkFrom1C = false`. This protects ERP-sourced data from accidental deletion.

Source: `TimeMgr.cs / DeleteUserException`

VR-007: Root Task Immutability

The root task (auto-created with the same name as the project) cannot be:

- **Deleted** — throws `DeleteRootProjectTaskException`
- **Renamed** — throws `ChangeRootProjectTaskException`

When a project is renamed, the root task is automatically renamed in the same transaction [GPT-5].

Source: `ProjectTaskMgr.cs:102-164`

VR-024: Assignment Lock When Filled [GPT-5]

Cannot change an assignment's **role** or **dates** when the timesheet for that period has already been filled. This prevents retroactive changes that would invalidate existing time entries.

Source: `ProjectAssignmentsMgr.cs / Change`

5. Calculation Rules

CR-001: Hours Per Day

Formula: `HOURS_IN_WORK_DAY = 8.0`

All time calculations assume an 8-hour standard workday. This constant is used throughout the system for capacity conversion, budget calculation, and working day computation.

Source: `Consts.cs:104`

CR-004: Days from Hours

Formula: `ProjectDays = TotalHours / 8`

Converts accumulated hours back to days for reporting purposes.

Source: `ProjectsCostReport.cs:55-60`

CR-005: Working Day Calculation

Formula: Working days = calendar days − weekends − branch holidays + WorkOff exception dates

The `GetWorkingDaysCount` method counts days where:

- Day is not Saturday or Sunday
- Day is not a branch-specific holiday
- OR day has a WorkOff exception (overrides non-working status)

Source: `TimeMgr.cs:248-270`

CR-006: Capacity to Hours Conversion [GPT-5]

Formula: `hours = capacityPercent × 8.0 / 100.0`

The `CreateCapacityConverter` method in `UserCalendar.cs` provides a function that converts a user's capacity percentage to daily available hours.

Source: `UserCalendar.cs / CreateCapacityConverter`

CR-007: Overtime Split [GPT-5]

Formulas:

- `InternalTime = Hours × k_internal`
- `ExternalTime = Hours × k_external`
- Null coefficients → 0 (BR-007)

This determines how overtime hours are split between internal (non-billable) and external (billable) categories. The split happens during the overtime save process.

Source: `TimeSheetByMonthMgr.cs`

CR-008: Week Number Calculation [GPT-5]

Algorithm: Custom ISO-8601-like week numbering

The system uses a **custom** algorithm for calculating week numbers, not .NET's built-in `Calendar.GetWeekOfYear`. Weeks start on Monday, and the algorithm handles year boundaries specially.

Source: `TimeMgr.cs / GetWeekNumber`

CR-012: Standard Hours Auto-Calculation [GPT-5]

Standard hours in Division Time Approval are **calculated** during insertion, not entered by the user. The calculation uses the user's capacity percentage and the number of working days in the period.

Source: `DivisionMgr.cs / InsertDivisionTimeApproval`

CR-013: Overtime Multiplier Aggregation [GPT-5]

Formulas:

- `extOvertime = SUM(hours × k_external)` across all overtime records
- `intOvertime = SUM(hours × k_internal)` across all overtime records

Used in `GetUserSpentHoursFromAssignments` to aggregate billable and non-billable overtime.

Source: `ProjectMgr.cs / GetUserSpentHoursFromAssignments`

---

6. Workflow Rules

WF-009: Filling Window [GPT-5 + NotebookLM]

Timesheet filling for a given month opens **3 working days before the month ends**. This is controlled by the configurable parameter `Configurator.DaysWhenTimeApprovalPeriodStarts`.

The method `TimeMgr.IsUserAbleToFillTimeSheet` checks this threshold to determine if the user can currently fill timesheets for the month.

Source: `TimeMgr.cs / IsUserAbleToFillTimeSheet`

WF-010: Previous Week Fallback [GPT-5]

If the current timesheet period has no tasks assigned, the system loads tasks from the **previous week** (WeekStart − 7 days), but only if those tasks are still accessible (`IsAccessible` check).

Source: `TimesheetController.cs / GetTimesheetHours`

WF-001: Approval Source Configuration

The time approval system can operate in two modes, toggled by the `project_time_based_on_WTS` flag:

- `true` → uses **TimeSheet data** (actual hours entered)
- `false` → uses **Assignment data** (planned hours from time frames)

Source: `ProjectTimeApproval.cs:28-84`

WF-006: Approval OR Logic [GPT-5]

The `DateIsApproved` method returns `true` if **either**:

- A **project-level** approval exists for the date, OR
- A **division-level** approval exists for the date

This is OR logic — both approval types independently lock the date. A date can be locked by project approval even without division approval, and vice versa.

Source: `TimeMgr.cs / DateIsApproved`

7. Calendar & Exception Integration

The timesheet domain depends heavily on the calendar and exception subsystem for determining working days, validating entries, and computing capacity.

8. Exception Types

Value	Name	Effect on Timesheet
1	SickLeave	Day not counted as working
2	DayOff	Day not counted as working; c
3	UnpaidAbsence	Day not counted as working; c
4	Vacation	Day not counted as **regular*
5	WorkOff	**Inverts** non-working day t
6	BusinessTrip	Requires active project (VR-0
7	PlannedVacation	Auto-deleted when actual vaca
8	OvertimeUndertime	Overtime/undertime tracking
9	CorporateActivity	Can overlap with PlannedVacat
10	CompensatedVacation	Compensated vacation tracking
11	Inactive	User inactive period
12	MaternityLeave	Extended leave
13	ApprovedAbsence	Can overlap with PlannedVacat
14	ChildCare	Child care leave; counted as

8. Approval Pipeline (Downstream)

The timesheet feeds into a multi-stage approval pipeline. This section summarises the downstream flow.

Pipeline Flow

```
Employee fills timesheet (TimeSheet3.aspx)
         │
         ▼
PM approves per project (ProjectTimeApproval.aspx)
  • Source: timesheet data OR assignment data (WF-001)
  • Requires SOW coverage (WF-011) [GPT-5]
  • Notifications only for projects with send_TA_notifications flag [GPT-5]
         │
         ▼
Division manager approves (DivisionTimeApproval.aspx)
  • Standard hours auto-calculated (CR-012) [GPT-5]
  • Blocked on frozen months (WF-002)
  • No duplicate approvals per user per month (WF-003)
         │
         ▼
Admin freezes month (MonthFreezing.aspx)
  • Requires ApproveAllDivisions permission (AC-009) [GPT-5]
  • Blocks ALL changes globally
         │
         ▼
ERP Export (ReportTimeApprovalForAxapta.aspx)
  • Stored procedure: pr_make_AXAPTA_report @end_month_date
```

9. Integration Points

Timesheet data flows to two external systems:

9.1 Axapta (Dynamics AX) ERP

- **Method**: SQL stored procedure `pr_make_AXAPTA_report`
- **Trigger**: Manual via `ReportTimeApprovalForAxapta.aspx`
- **Permission**: `AgainstReports().ViewAxapta()`
- **Data**: Approved time entries aggregated by month

9.2 DocuWare Document Management

- **Method**: REST API via .NET SDK
- **Trigger**: Scheduled job `SendInvoicesOfRemoteHourlyUsersJob`
- **Data**: Time approval invoices for remote hourly employees
- **Known issue** [GPT-5]: `OrganizationName = "First Line Software"` is **hard coded**; must be configurable in reimplementation

9.3 BDW (Big Data Warehouse) [GPT-5]

- **Method**: SQL Linked Server via `OPENQUERY` to `BDW_MIGRATION_SOURCE`
- **Data**: Overtime data sync using event sourcing with `I/U/D` event codes
- **Composite key**: `TabNumber:BranchGroup:ProjectId:Mark`