timesheeting system architecture document

Thomas HOULLIER ⟨pro@houllier.net⟩

PRJ1-SAD1-v1.0 .1 – February 10November 28, 2024

Abstract

This is the system architecture document for the timesheeting project. It proposes an architecture for a system in answer to the timesheeting specification.

Revision History

Revision	Date	Author(s)	Description
1.0	10FEB2024	TH	Creation
1.1	28NOV2024	TH	Aligned the architecture with the updated PRJ1-SPE1-v1.1.

Applicable documents

Index	Title	Reference	Revision	Author
AD1	timesheeting specification document	PRJ1-SPE1	v1.0 .1	Thomas HOULLIER

Document distribution

Contents

Acronyms

CI Continuous integration
CLI Command line interface
DB Database
GUI Graphical user interface
UI User interface

1 Introduction

timesheeting is a software project for creating, managing and reporting timesheet data. The present document is in answer to the project's user specification document [AD1].

The document is organized as follows. We outline the overall architecture briefly in Section 2. The system architecture is decomposed into subsystems and presented in Section 3. Finally, the user requirements [AD1] are dispatched to the subsystems in Section 4.

2 Overall architecture

The proposed timesheeting software is a desktop ~~Graphical user interface (GUI)~~ linux application. The software embeds a Database (DB) for saving timesheet data ~~. It~~ and software state. The software is configured through a user dotfile. It ~~includes a utility to make a backup of the database. It~~ also includes timesheet data export capabilities to an interoperable format. A ~~state file is used for ergonomy. A logging system is proposed~~logging system to a file is included.

The software is distributed through source releases. The associated documentation is distributed via a documentation distribution channel. The software and documentation releases are signed. We propose testing pipelines. The user can file bug reports to an issue tracking system.

3 System architecture diagram

The proposed system is decomposed into the following subsystems. We distinguish between the subsystems active while the user interacts with the software

(online subsystems), and the auxiliary susbsystems (offline subsystems). We outline the function of each subsystem and provide a diagram for illustration.

3.1 Online subsystems

The software comprises the following online subsystems,

- The **~~GUI~~User interface (UI)** displays the ~~UI screens~~ GUI screens and makes the Command line interface (CLI) available to the user. ~~It allows the user to inputinformation into the software and interact~~The GUI and CLI both allow user input.
- The **Core logic** handles the manipulation of timesheet data for presentation to the user. It decouples the ~~GUI~~ UI subsystem from the rest of the application.
- The ~~**Settings**~~ **Configuration manager** loads the user configuration file, and propagates the settings to the application.
- The **Exporter** is in charge of generating the export file for timesheet data.
- The **DB backend** interacts with the DB. ~~It decouples the database operation from the rest of the application.~~
- The ~~**Backup manager** creates and restores backup files for the current application state.~~
- ~~The~~ **Logger** records the application logs.

Fig. 1 illustrates the interaction between these subsystems.
The data files involved in the operation of the software are,

- The **User configuration file** allows the user to configure the application. This is a text file the user edits.
- The ~~**User state file** records some of the application states for ergonomy and interaction. It does not contain any critical data such as timesheet data.~~
- ~~The~~ **Logs** record application events for debugging purposes.
- The **Database** allows the application to store and retrieve ~~critical data~~ data contents. It is persistent.
- The **Export file** is a user-readable file generated on-demand from the timesheet data.
- ~~The **Backup** is an archive recording the critical application data, for saving the data elsewhere, or transfering between systems. It allows restoring the database.~~

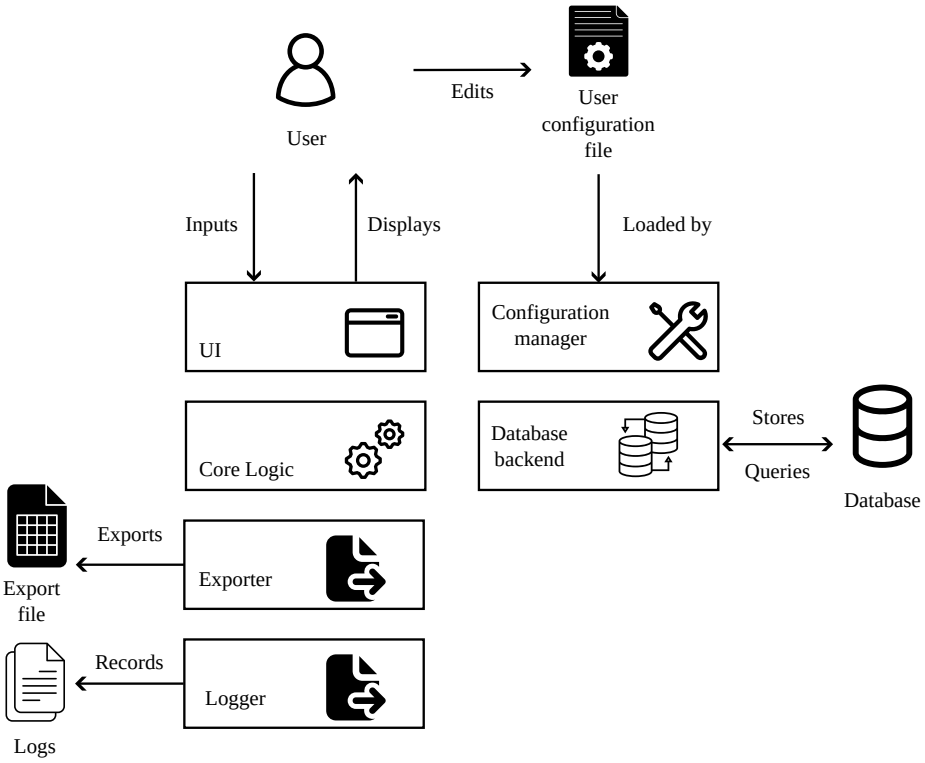

Figure 1: System architecture diagram.

3.2 Offline subsystems

The software comprises the following offline subsystems,

- The **Versioning** system tracks the state of both the software source and documentation source.
- The **Software distribution** system manages the release of software source to the user.
- The **Build** system generates a binary from the software source.
- The **Automated testing** comprises the tests that are run automatically for every software release. It is part of the Continuous integration (CI) pipeline.
- The **Manual testing** comprises the tests performed by the **Tester**, manually, for every software release. It is part of the tester pipeline.
- The **Documentation generator** generates the documentation releases from the documentation source.
- The **Documentation distribution** manages the release of documentation reports.
- The **Signature system** authenticates the released source and released documentation as coming from the author.
- The **Issue tracking** allows the *user* to file bug reports.

Fig. 2 illustrates the interaction between these subsystems.
There are three build pipelines in parallel,

- The **User** pipeline: the *user* receives the *released source*, and uses the *build* system to generate a binary. The *user* then uses the binary.
- The **CI** pipeline: The software source for every version are automatically sent to a CI system, where the source are built into a binary. Assuming the build was successful, automated tests are then run on the binary. The results are reported publicly in the software distribution system.
- The **Tester** pipeline: The software source is built by the tester using the build system. The binary is then tested against a set of manual cases.

4 Requirements dispatch

We assign responsibility for the requirements in [AD1] to the various subsystems described in the system architecture (Section 3). This dispatch is presented in Tab. 1. Some requirements are attributed to a *General* system, when they cannot be attributed to a single subsystem. This dispatch is not *strict*, it only indicates

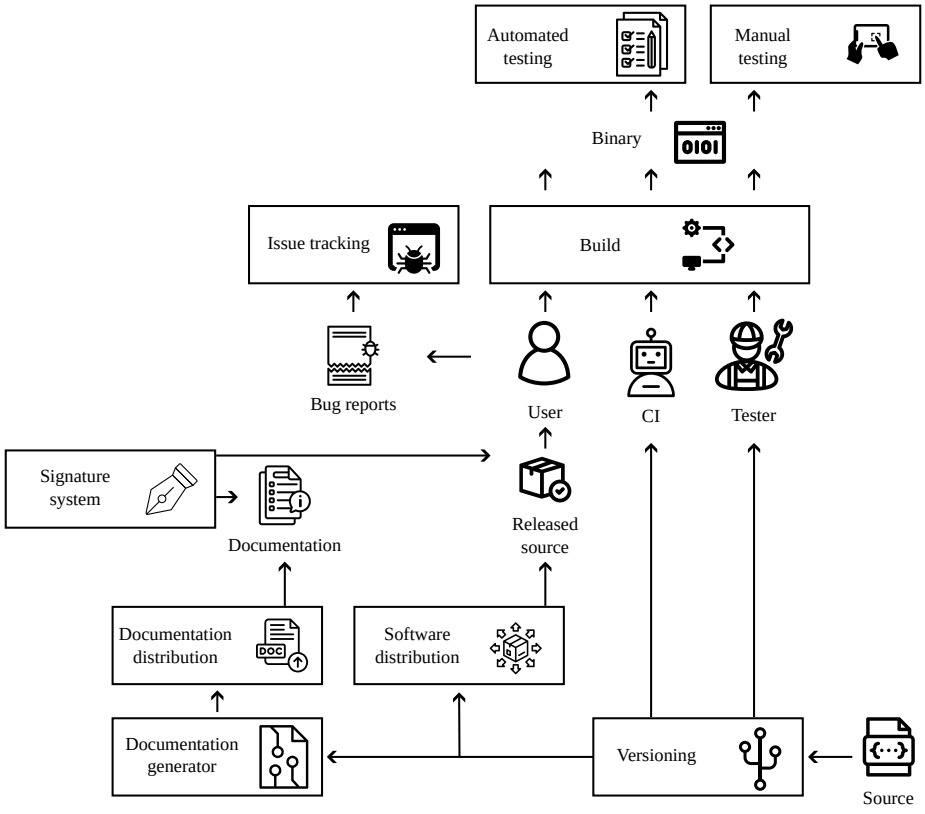

Figure 2: Offline subsystems diagram.

in which subsystems we expect to find the most requirements answering to a given user requirement.

Table 1: Requirements responsibility dispatch to subsystems.

Req. ID	Req. name	Responsible subsystem
R-UHI-010	Adding hierarchy items	~~GUI, Core logic~~ UI, DB backend
R-UHI-020	~~Removing~~ Inactivating hierarchy items	~~GUI, Core logic~~ UI, DB backend
R-UHI-030	Editing hierarchy items	~~GUI, Core logic~~ UI, DB backend
R-UHI-040	~~Restoring deleted~~ Activating hierarchy items	~~GUI, Core logic~~ UI, DB backend
R-UHI-050	Project ~~removal~~ inactivation effect	~~Core logic~~ UI
~~R-UEI-010~~ R-UHI-060	~~Adding entries~~ Removing hierarchy items	~~GUI, Core logic~~ UI, DB backend
~~R-UEI-020~~ R-UEI-010	Adding entries ~~through stopwatch~~	~~GUI, Core logic~~ UI, DB backend
~~R-UEI-030 Adding entries manually GUI, Core logic~~ R-UEI-040	Removing entries	~~GUI, Core logic~~ UI, DB backend
R-UEI-050	Editing entries	~~GUI, Core logic~~ UI, DB backend
R-UGL-010	UI screens breakdown	~~GUI~~ UI
R-DES-010	Daily entries	~~GUI~~ UI
R-DES-020	Day selection	~~GUI~~ UI
~~R-DES-025 Day selection format GUI~~ R-DES-030	Display entries of the day	~~GUI~~ UI

R-DES-040	Running daily total	~~GUI~~ UI
~~R-STP-010 Stopwatch in use GUI R-STP-020 Running stopwatch time GUI R-STP-030 Stopwatch only on current day Core logic R-ENI-010 Entry metadata prefill GUI R-ENI-020 Entry metadata suggestion GUI~~ R-ENI-030	Entry metadata hierarchy search	~~GUI~~ UI
R-ENI-040	Entry metadata hierarchy coherence	~~Core logic~~ DB backend
R-HIS-010	Hierarchy items	~~GUI~~ UI
R-HIS-020	Hierarchy items display	~~GUI~~ UI
R-GUI-010	Keyboard usage	~~GUI~~ UI
R-LDC-010	Entry identification	DB backend
R-LDC-020	Entry metadata	DB backend
~~R-LDC-030 Company identification DB backend R-LDC-040 Company metadata DB backend~~ R-LDC-050	Project identification	DB backend
R-LDC-060	Project metadata	DB backend

R-LDC-070	Task identification	DB backend
R-LDC-080	Task metadata	DB backend
R-TIM-010	Time standard	Core logic
R-TIM-020	Time reference	Core logic
R-TIM-030	Time zones	Core logic, ~~Settings~~ Configuration manager
R-TIM-040	Time resolution	Core logic
R-SAV-010	Save	DB backend
R-SAV-020	Transparent save	~~DB backend~~ UI
R-SAV-030	Timesheet save resolution	~~DB backend~~ UI
R-SAV-031	Hierarchy items save resolution	~~DB backend~~ UI
~~R-SAV-040 Save status GUI~~ R-SAV-050	Switch save profile	~~Settings~~ Configuration manager
R-SAV-060	Starting save profile	~~Settings~~ Configuration manager
R-BAK-010	Backup	~~Backup manager~~ General
R-BAK-020	Backup restore	~~Backup manager~~ General
R-BAK-030	Backup completeness	~~Backup manager~~ General
R-BAK-040	Backup conciness	~~Backup manager~~ General
~~R-BAK-050 Backup timestamp Backup manager~~ R-BAK-060	Backup naming	~~GUI~~ General
R-BAK-070	Backup location	~~GUI~~ General
R-DEX-010	Timesheet export	Exporter
R-DEX-020	Export naming	~~GUI~~ UI
R-DEX-030	Export location	~~GUI~~ UI

~~R-DEX-040 Export tool screen GUI~~ R-DEX-050	Export time period	~~GUI~~UI, Exporter
R-ACC-010	Single user	Core logic
R-ACC-020	Synchronization across systems	General
R-ACC-030	Company segregation	Core logic
R-ACC-040	Data confidentiality	DB backend
R-ACC-050	Offline operation	Core logic
R-ACC-060	Interface language	UI
R-ACC-070	Documentation language	General
R-ACC-080	Offline operation	General
R-ENV-010	Target hardware	General
R-ENV-020	Target OS	General
R-ENV-030	Target OS version	General
R-ENV-040	Target graphical environment	~~GUI~~ UI
R-PER-010	Memory footprint	General
R-URE-010	Durations display format	~~GUI~~ UI
R-URE-020	Day duration definition	UI
R-RPT-010	Project totals	~~GUI~~ UI
R-RPT-020	Project totals time period	~~GUI~~ UI
R-RWT-010	Weekly report	~~GUI~~ UI
R-RWT-020	Weekly report daily totals	~~GUI~~ UI
R-RWT-030	Weekly report weekly totals	~~GUI~~ UI
R-RWT-040	Weekly report running week	~~GUI~~ UI
R-RWT-050	Weekly report week selection	~~GUI~~ UI

~~R-RWT-070 Weekly report timesheet export Exporter~~ R-LOG-010	User data interaction logging	Logger
R-LOG-020	Log file location	Logger
~~R-LOG-030~~ R-LOG-025	Log ~~depth~~ cleanup period	Logger
R-LOG-040	Log cleanup	Logger
R-LOG-050	Log cleanup schedule	Logger
R-LOG-060	Log readability	Logger
R-LOG-070	Log accessibility	Logger
R-QUA-010	Version report	~~GUI~~ UI
R-QUA-020	Save data validation	Core logic
R-QUA-025	Save data action	DB backend
R-QUA-030	Release signature	Signature system
R-QUA-040	Single repository	Versioning
R-QUA-050	Bug tracker information	General
R-TES-010	Automated build test	Automated testing
R-DOC-010	Development documentation	General
R-DOC-020	User manual	General
~~R-DOC-030 Keyboard cheatsheet General R-DOC-040 Keyboard cheatsheet conciness General~~ R-DOC-050	Software build instructions	General
R-DOC-060	Documentation build instructions	Documentation generator
R-DOC-070	Matrix of conformity	General
R-DOC-080	Architecture and design document	General
R-REL-010	Software version format	General
R-REL-020	Release notes	General

R-REL-030	Release notes granularity	General
R-REL-040	Release notes publication	Documentation distribution
R-REL-050	Documentation release	Documentation distribution
R-REL-060	Build dependencies	Build
R-REL-070	Release format	Build
R-DEP-010	Installation script	Build
R-DEP-020	Uninstallation script	Build
R-LIC-010	Source code license	General
R-LIC-020	Documentation license	General