Hotaru

A Rust web framework. Project overview and contributor guide.

FDS-PMINE & Project-StarFall

2026
Field of Dreams Studio

Table of content

About this deck

What this is.
A project overview of Hotaru, a Rust web framework built by FDS. It covers what Hotaru already does, what we plan to build next, and where new contributors can plug in.

Who it is for.
Anyone curious about joining the project, regardless of Rust experience. We work without fixed deadlines or required hours.

What we are looking for.
Long-term contributors: coders, writers, community organisers.

1 What Hotaru Is

The big picture

The aim.
Make Rust web servers as easy to write as scripting languages. Multi-protocol from one framework (HTTP, WebSocket, MQTT), with Rust's safety and speed.

Why we built it.
FDS needed a web framework for its own projects. We built one we wanted to use, then shared it.

Who maintains it.
One of several projects in FDS's portfolio. Maintained jointly by FDS-PMINE (applied Rust research) and Project-StarFall.

Hotaru in one line

Small, sweet, easy.
A full-stack web framework for Rust applications, focused on simplicity over ceremony.

Already shipping.
Version 0.8.2 on crates.io, the public Rust package registry.

Free for anyone to use.
Released under the MIT license. Fork it, learn from it, build on it.

Speaks more than HTTP.
HTTP, HTTPS, WebSocket. MQTT in active development.

2 Tech Stack

Why Rust

Fast.
As fast as C and C++, without the common crashes.

Safe.
The compiler catches whole classes of bugs (memory bugs, data races) before your program runs.

In real use.
Discord, Cloudflare, Microsoft, Meta, and the Linux kernel.

Wide-reaching.
Once you know Rust, you can work on backends, embedded devices, browsers, and operating systems.

Hotaru "Hello World"

Declare the server.

```
use hotaru::prelude::*;
use hotaru::http::*;

LServer!(
    APP = Server::new()
        .binding("127.0.0.1:3003")
        .single_protocol(ProtocolBuilder::new(
            HTTP::server(HttpSafety::default())))
        .build()
);
```

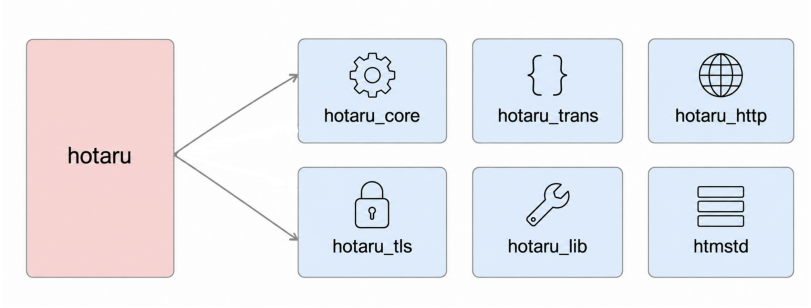
Hotaru "Hello World" (Cont'd)

Add an entry point and a route.

```
#[tokio::main]
async fn main() {
    APP.clone().run().await;
}

endpoint! {
    APP.url("/"),
    pub index<HTTP> {
        text_response("Hello, Hotaru!")
    }
}
```

The crate ecosystem



Modular workspace.
Seven published crates on crates.io: `hotaru`, `hotaru_core`, `hotaru_trans`, `hotaru_http`, `hotaru_tls`, `hotaru_lib`, `htmstd`. Each is small enough to read and own.

3 Current Achievements

Capabilities today

Web servers.
Serve HTTP and HTTPS sites with a few lines of code.

Templates and forms.
Render HTML with the Akari engine. Handle forms, file uploads, sessions, and cookies.

Real-time.
Chat-style and live data applications over WebSocket.

Multiple protocols, one server.
Mix HTTP, WebSocket, and your own protocols inside the same server.

Real research use: HCR at PolyU

Figure placeholder: `static/hcr_overview.png`

Built on Hotaru.
The Hair Cutting Robot (HCR) research project at The Hong Kong Polytechnic University runs on the Hotaru framework. A cross-institutional collaboration with IITK India and LNTU China.

4 What Is Left to Build

MQTT and the IoT story

What MQTT is.
The standard messaging protocol for the Internet of Things: sensors, robots, smart devices.

What we are adding.
A native MQTT module in the Hotaru ecosystem (`hotaru_mqtt`), already in development on the `draft` branch on GitHub.

First-class support.
Other Rust web frameworks rely on separate MQTT libraries. Hotaru integrates MQTT as a peer protocol to HTTP at the framework level.

Status.
Active development. Newcomers can help shape the API and the test suite.

Roadmap

HTTP/2, HTTP/3, WebSocket improvements.
Modernise the HTTP transport layer. Keep extending WebSocket.

no_std support.
Make Hotaru's core usable in embedded contexts (no operating system, no standard library).

HTTP and MQTT feature expansion.
Steady widening of both protocol stacks.

Maintenance of the codespace.
Tests, examples, error messages, dependency hygiene across all seven crates.

5 The Team

Two teams, one project

FDS-PMINE.
FDS's applied Rust research group. Owns Hotaru's runtime (core engine, connections, HTTP, MQTT, standard middleware). Broader work spans memory-safe systems, plugin architecture, and clinical AI.

Project-StarFall.
Owns Hotaru's macro and language layer, the syntax that makes the framework easy to use.

Standards.
The expectations for every contributor are listed in the FDS Constitution. Both teams follow them.

AI honesty contract

We use AI tools and we are open about it. Every module declares its AI tier.

Author-Owned.
Written by humans. AI does not show through.

Human-Led.
Humans design the load-bearing pieces. AI fills boilerplate.

Co-Authored.
AI is a real collaborator. The human still defends every line.

No matter the tier, every contributor must understand what they ship.

6 How You Can Join

How joining works

No fixed schedule.
You set your own pace. No deadlines, no required hours per week.

Pick from a shared pool of work.
A running list of tasks (docs, code, tutorials, social). Take what calls to you. Drop one, try another.

Ask for help on your first ones.
If you want a walkthrough on your first contribution, ask. Someone will sit with you.

Full support on the toolchain.
Git, GitHub, LaTeX, the Rust toolchain. If you have not used them, we walk you through.

Five ways to contribute

Pick what fits your skills and your time. You can switch roles later.

- **Core development.** Write Rust code: build the inner language, the template engine, the bridge to Rust.
- **Language theory.** Help shape the inner language. Think about correctness, types, and the guarantees the language should give its users.
- **Documentation.** Write tutorials, examples, a guide for first-time contributors.
- **Social and community.** Run the website, post updates, welcome new members, organise small events.
- **Use and feedback.** Build something with Akari (or Hotaru) and tell us what hurts.

Two ways to start

Open a small pull request.
You do not need team membership to land a small PR. Pick something from the pool, send it, get reviewed. We help if you ask.

For long-term membership: FDS0.
A four-week, mentor-paired onboarding curriculum. Covers identity, rules, Git, LaTeX, AI agents, the FDS workflow. Ends with one small real FDS contribution. After it, you are a recognised FDS member.

Your first Hotaru app, in three commands

```
# 1. Install Rust (one-time setup)
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh

# 2. Install the Hotaru CLI
cargo install hotaru

# 3. Create and run a new project
hotaru new my_app
cd my_app && cargo run
```

A working website appears at http://127.0.0.1:3003.

How to apply to FDS

Step 1.
Read the FDS Constitution at https://doc.fds.moe. Available in English, Simplified Chinese, Traditional Chinese, and Japanese.

Step 2.
Fill in the FDS application form. Direct link in "Where to find us" below.

Step 3.
Reach out via QQ group 590328937, email redstone@fds.moe, or Discord (link below). An admin replies within three days.

The review is about your time and your ideas, not your technical skill.

Where to find us

Project hubs.
FDS at https://fds.moe.
FDS-PMINE at https://pmine.rs.
Hotaru at https://hotaru.rs.
Community writing at https://fds.rs.

Source and docs.
Code at https://github.com/Field-of-Dreams-Studio/hotaru.
MQTT at https://github.com/fds-pmine/hotaru_mqtt.
API docs at https://docs.rs/hotaru.

Where to find us (Cont'd)

Email and chat.
Email: redstone@fds.moe.
QQ group: 590328937.
Discord: discord.gg/Y6b9KRUCux.

Sign-up form.
Office Forms (click here). The form is also linked from the FDS Constitution at https://doc.fds.moe.