RF-3 - Software Architecture Specification

CTRL ALT Elite

Kevin Ekart / Ethan Gray / Laken Hollen / Huynh Le

October 17, 2025

Software Architecture Specification

Overview

- The system will use the Client-Server architecture. So, the client will be responsible for some tasks, and one or more servers will be responsible for others, and the two will communicate over the internet. The server will be responsible for more work than the client, though.
- Some of the tasks the client will be responsible for include interacting with the app's interface, changing data as needed, and submitting data to the server. Most of the core functionality of the app will be run on the client side, with the data associated with those functionalities being transmitted to the server.
- Some of the tasks the server will be responsible for include authenticating clients, storing any data the clients submit, and responding to any other requests from the client. The server will mostly be responsible for handling the data and logic flow.

Subsystem Decomposition

Database Server

- Securely store all the data submitted by clients
- Create and maintain logs so admins can see all interactions with stored data and who did them

Application Server

- Handle the app's logic and data flow
- Transmit data from the client to the database

Task Board

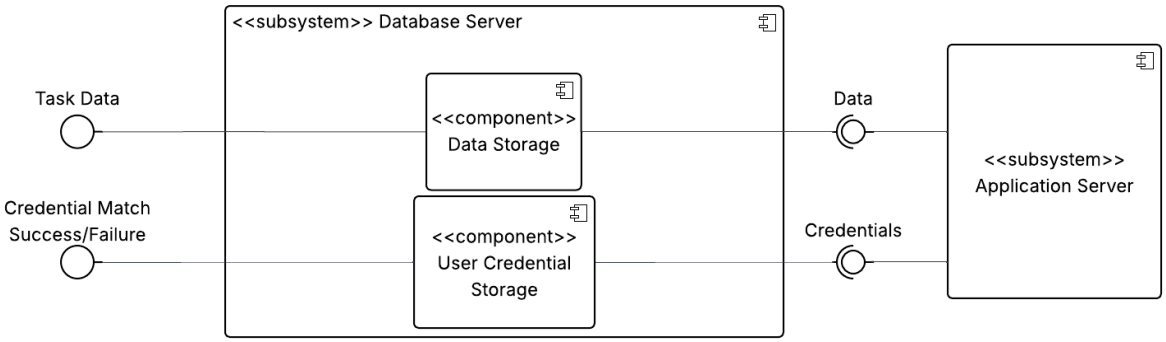
- Display the task board data from the database
- Transmit data to the database as changes are made to new and existing tasks

User Authentication Service

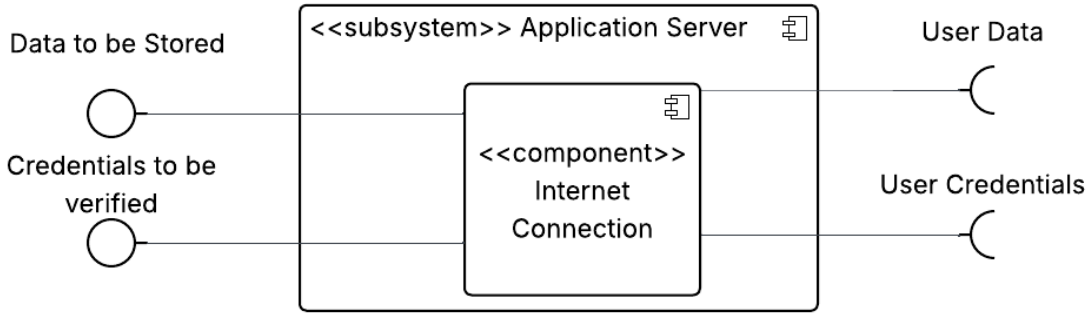
- Correctly authenticate users
- Communicate with the database to validate credentials
- Ensure that users have the correct permissions

Component Diagrams

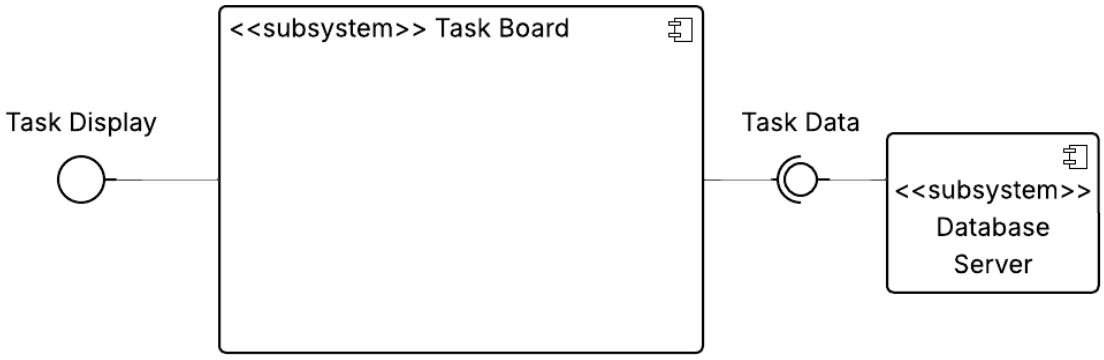
Database Server



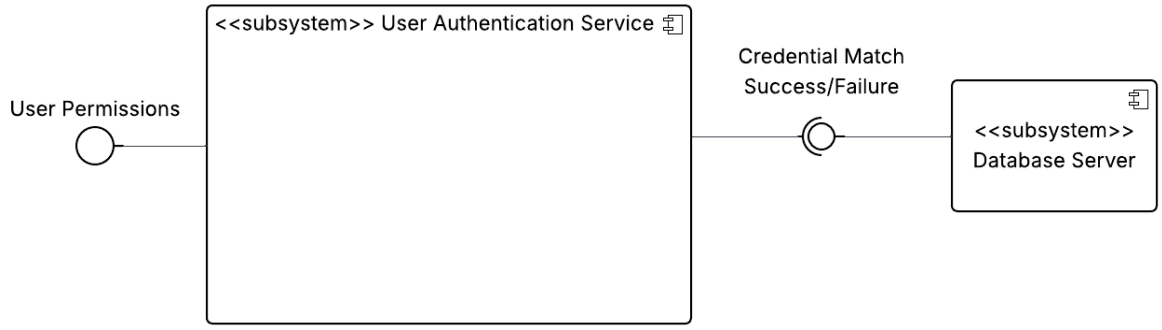
Application Server



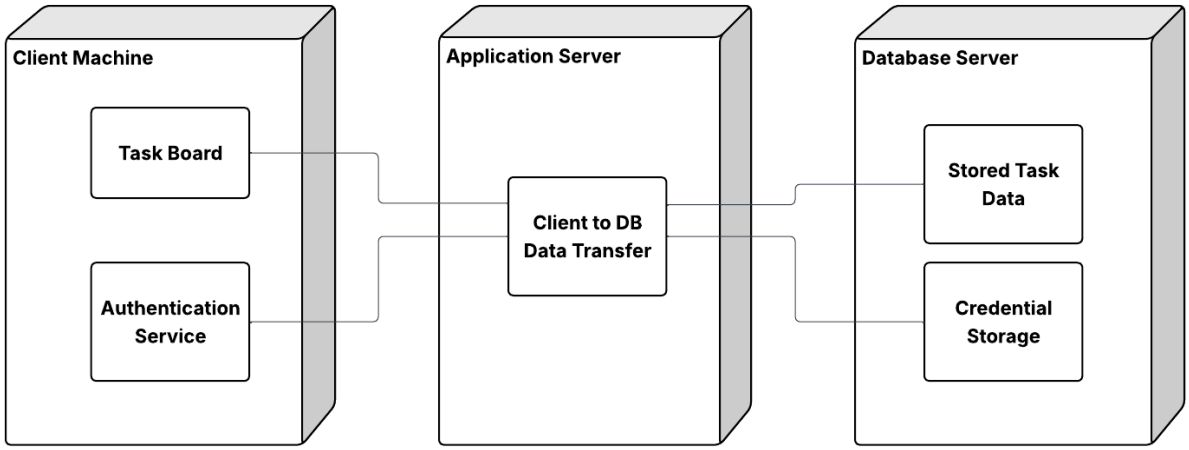
Task Board



User Authentication Service



Hardware/Software Mapping



- The client is responsible for displaying the task board and using the authentication service. Both require communication with the database server to work properly.
- The application server's primary function is to transfer data from the client to the database server.
- The database server is responsible for storing user credentials and task data and transmitting them to the client when needed.

Persistent Data Management

Objective

Explain what data we store, how it's organized, and who can access or change it—in a simple, secure, and team-friendly way.

What Data Do We Store?

We permanently save only what's needed to run the task board:

- Users: name, email, secure password, role
- Boards: project name, owner
- Lanes: columns like "To Do", "In Progress", "Done"
- Tasks: title, description, assignee, due date, which lane it's in
- Comments: messages on tasks
- Activity Log: who did what and when ("Moved task to Done")
- Board Members: which users can access which boards

How Is Data Organized?

- Users
  - user_id
  - name
  - email
  - password_hash
  - role
- Boards
  - board_id
  - name
  - owner_id
- Lanes
  - lane_id
  - board_id
  - name
  - order
- Tasks
  - task_id
  - lane_id
  - title
  - assignee_id
  - due_date
- Comments
  - comment_id
  - task_id
  - author_id
  - content

- Activity_Log
    - log_id
    - user_id
    - board_id
    - action
    - target
- Users_Boards
    - user_id + board_id → who can access which board
- Users have Boards
- Boards have Lanes
- Lanes have Tasks
- Tasks have Comments and Activity Logs

Security

- Authentication: Proving You're You Sign up/login with email + password
    - Passwords are hashed with bcrypt (never stored plainly)
- All communication uses HTTPS for safety

User Roles & Permissions

Role	What They Can Do
Admin	Manage all boards, users, and data
Project Manager	Full control of boards they own (create lanes, add members, edit tasks)
Team Member	View boards they're invited to, create/edit their own tasks, comment, move tasks Each role has specific permissions to perform actions We store data in a secure database

- Users log in to access boards and tasks
- Roles determine what actions users can perform
- We prioritize security to keep data safe

Global Software Control

Global Control Flow

The database adopts an event-driven architecture in which server responses and requests are triggered in real time by user interactions.

- **Client-side behavior:** User events like click, drag, or form submission invoke UI action.
- **Synchronous writes:** The interface uses REST API calls (e.g., POST /tasks, PATCH /tasks/{id}), writing the data directly to the database as the single source of truth.
- **Asynchronous fan-out**: Once the server successfully commits, it pushes the updates via WebSocket channels so that the connected clients remain in sync.
- **Concurrency model**: The REST request is dealt with in atomic terms. WebSocket broadcasts contain the version and timestamp values such that the clients skip late or duplicate messages.
- **Failure handling:** In case the transaction fails, the client reverts any optimistic changes and returns an error without any broadcast event being published.

It offers rapid feedback to the user with the promise of consistency across all the affiliated sessions.

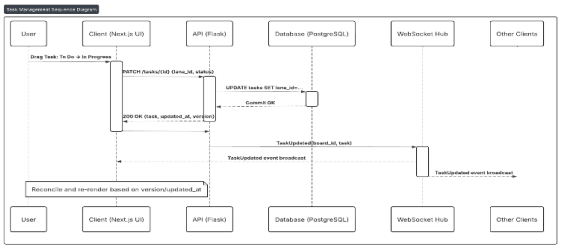
Key Interactions (Sequence Diagrams)

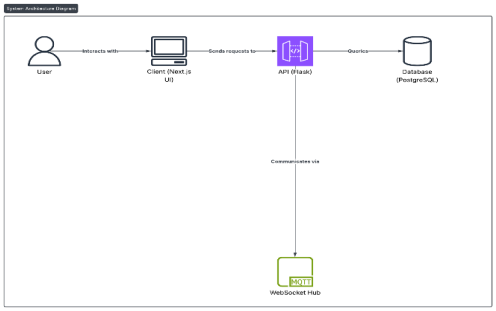
The following diagrams show the system's runtime exchanges among the user, client, API, database, and WebSocket hub.

Move a Task Between Lanes

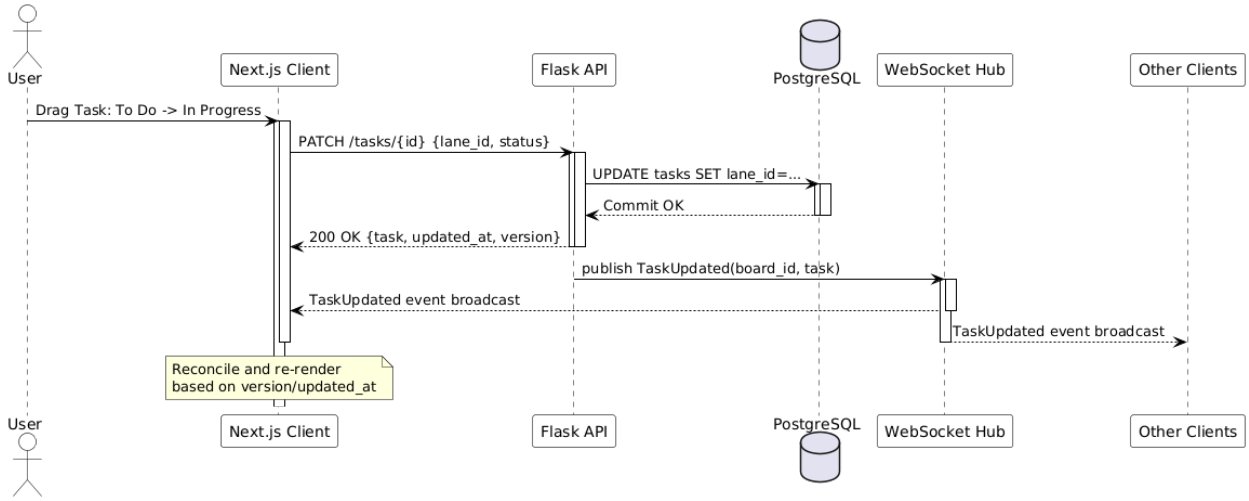
When a user drags a task to a new lane, the client sends a PATCH request to update the task's status. The backend commits the change and publishes an event that refreshes all connected clients.

Figure 1. Move a Task Between Lanes





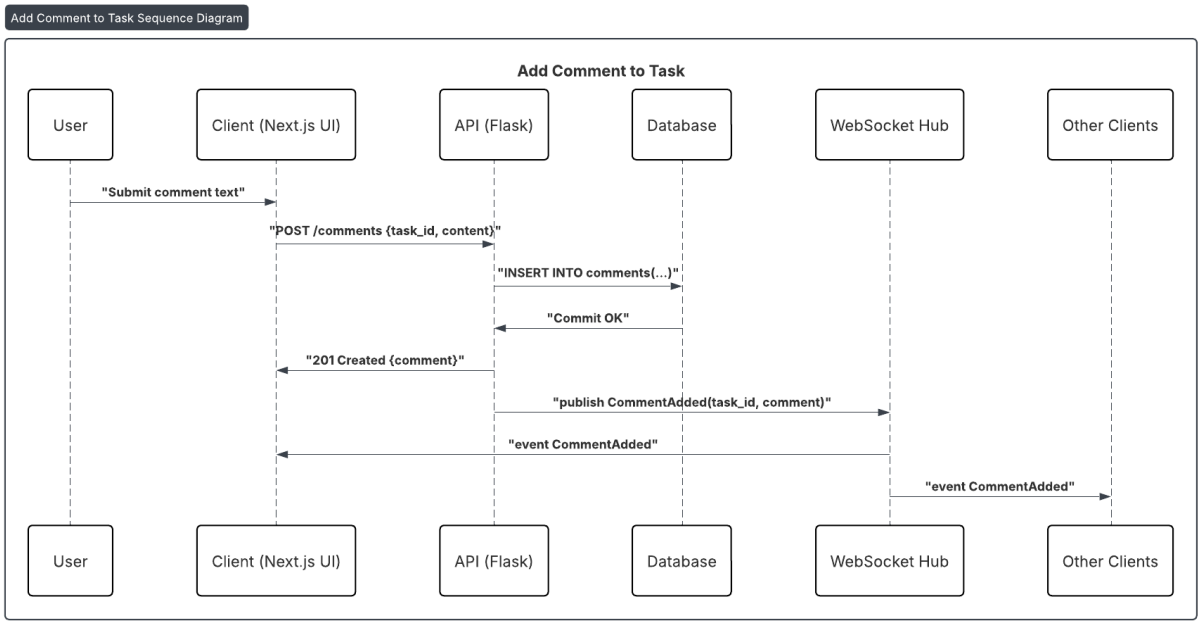
Additional Diagram:



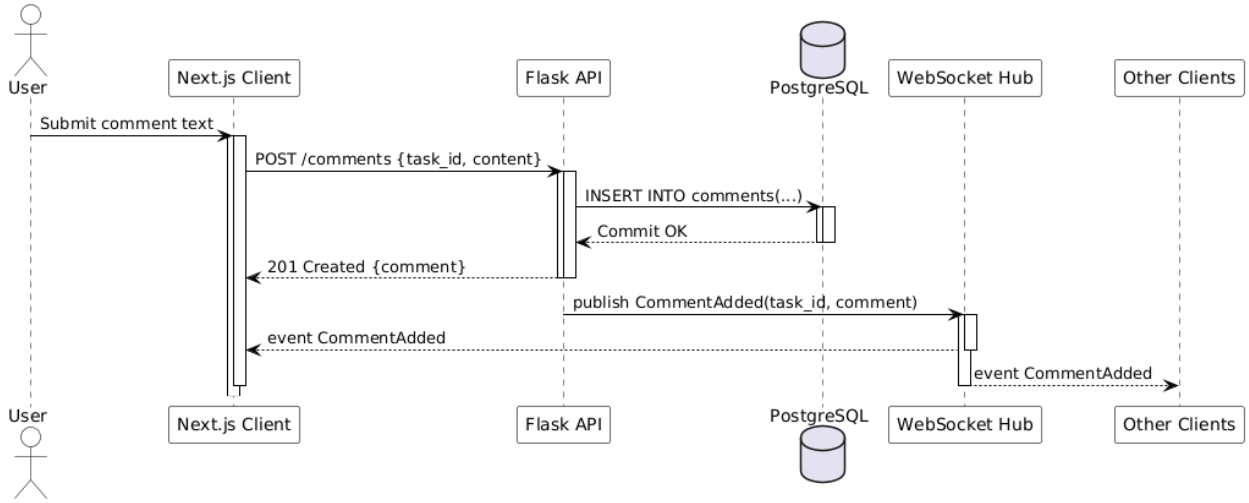
Add Comment to a Task

When a comment is submitted, a POST request is initiated. After the database saves the comment, the server returns a confirmation, and the update is sent through the WebSocket hub.

Figure 2. Add Comment to Task



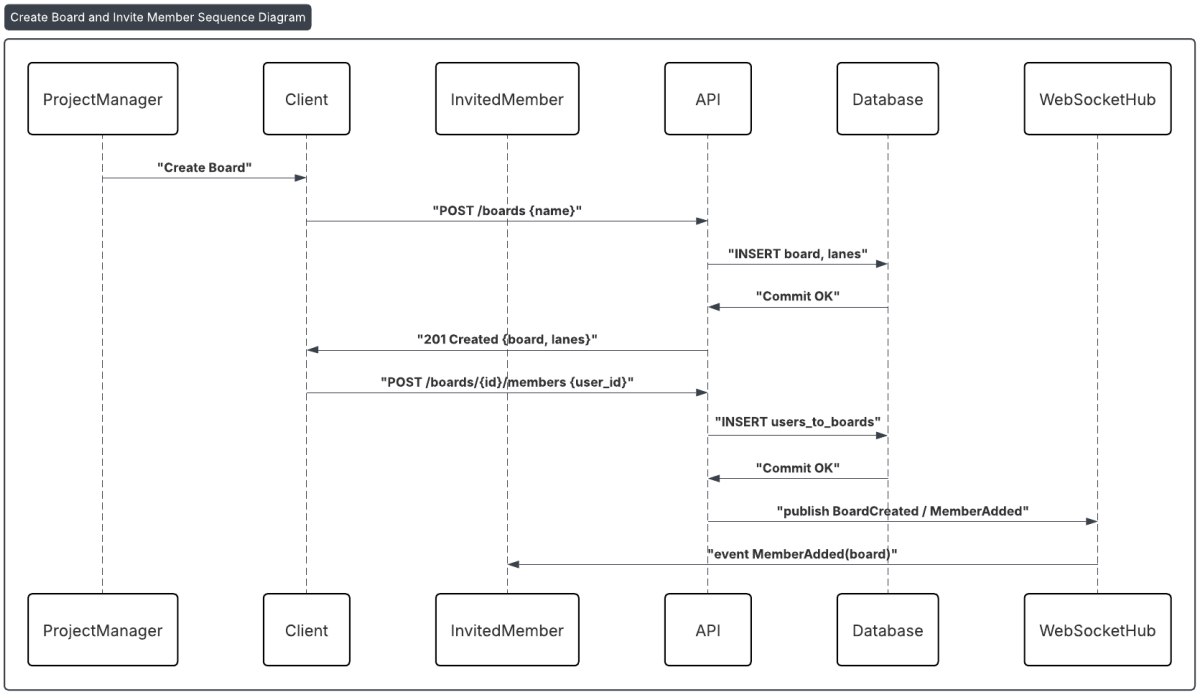
Additional Diagram:



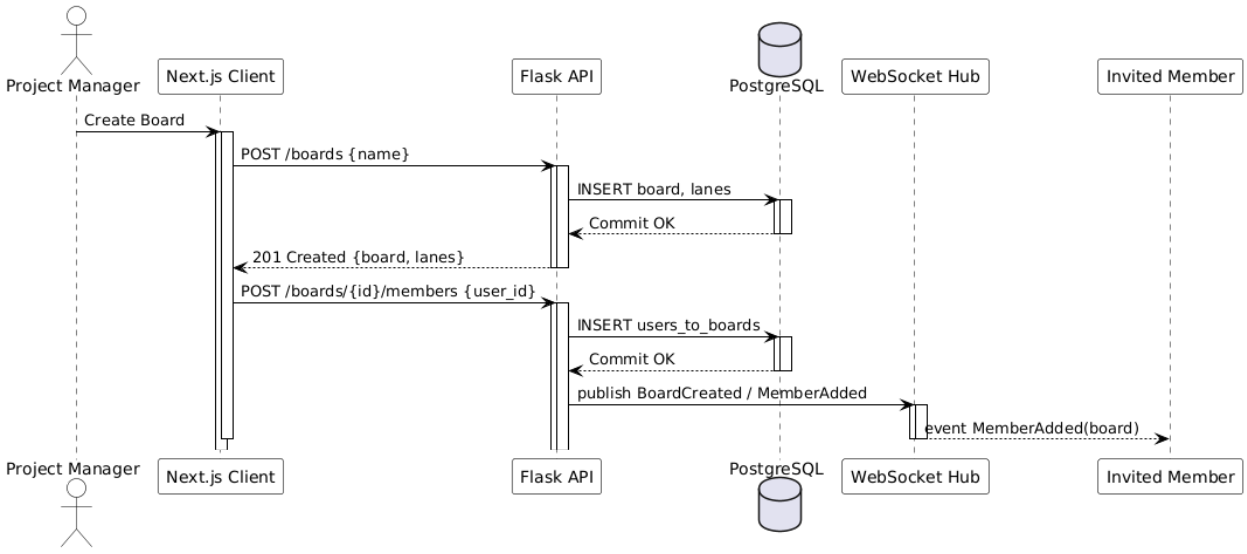
Create a New Board and Invite a Member

A project manager makes a board and adds people with two API calls, one after the other. Once both API calls are committed the system sends notifications to the newly added members.

Figure 3. Create Board and Invite Member



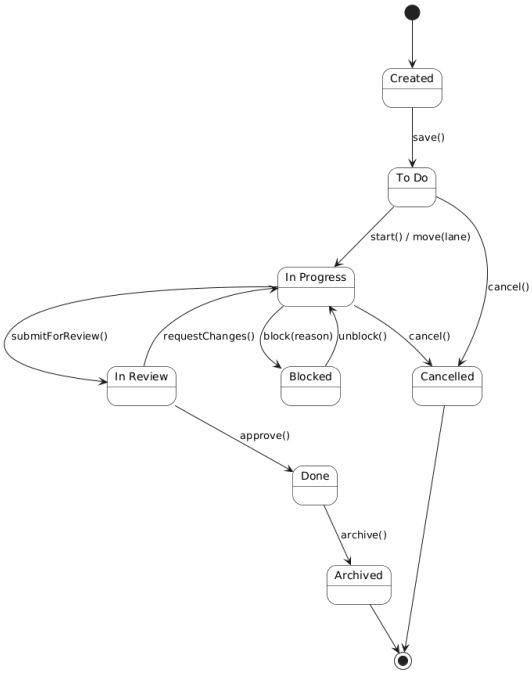
Additional Diagram:



Task Lifecycle

The diagram below shows the progress of a task through its various phases when someone creates the task, someone works the task, a reviewer reviews the task, the task gets completed, or the task gets cancelled. Each progression records the actor, the time, and the action associated with that task.

Figure 4. Task Lifecycle



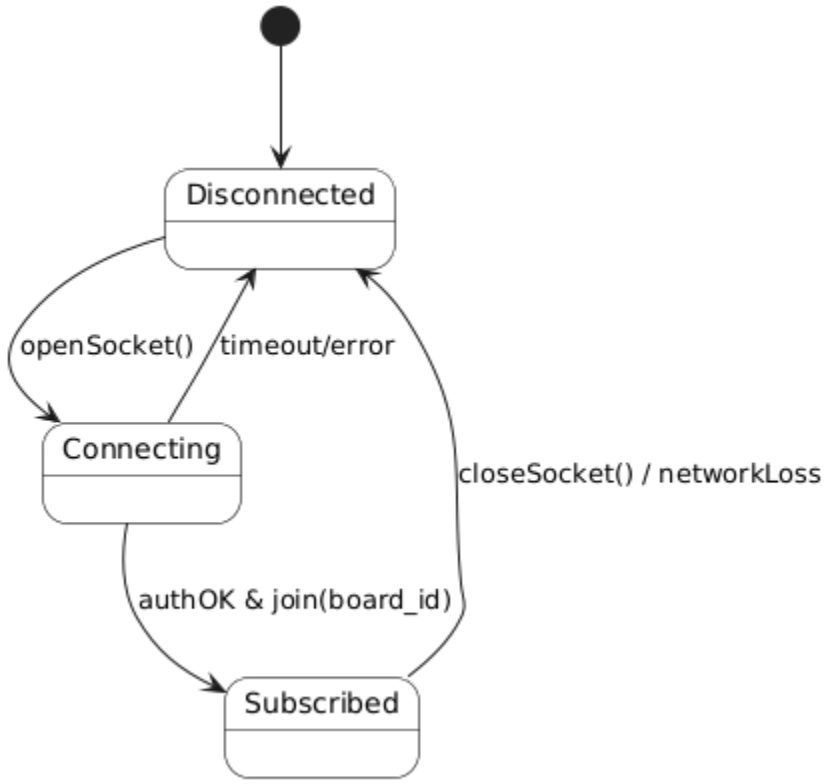
Key Transitions

- save() — moves a task from *Created* to *To Do*
- start() or move(lane) — moves a task into *In Progress*
- block(reason) / unblock() — toggles between *Blocked* and *In Progress*
- submitForReview() / requestChanges() / approve() — covers review flow
- archive() — moves a completed task to *Archived*
- cancel() — terminates a task from *To Do* or *In Progress*

Board Session / Realtime Presence

The following state machine shows how a client session transitions during WebSocket activity.

Figure 5. Realtime Presence



States and Transitions

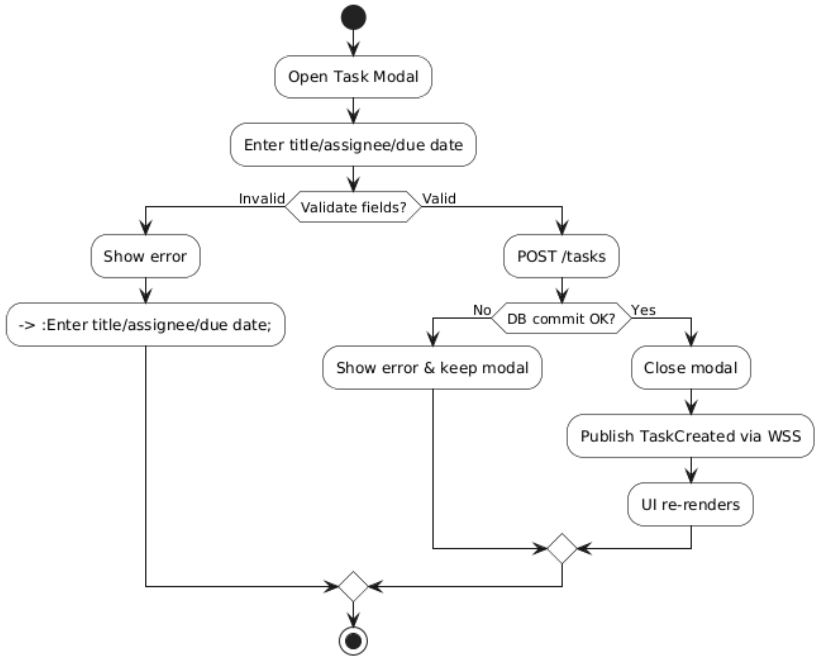
- Disconnected → Connecting: as soon as the WebSocket connection opens
- Connecting → Subscribed: once authentication is successful
- Connecting → Disconnected: when an error or a timeout occurs
- Subscribed → Disconnected: when the session is closed by the user or the network is lost

Workflows (Activity Diagrams)

Create Task Workflow

This workflow describes the steps taken to add a new task, validate the values provided, commit the task to the database, and notify the connected clients of change.

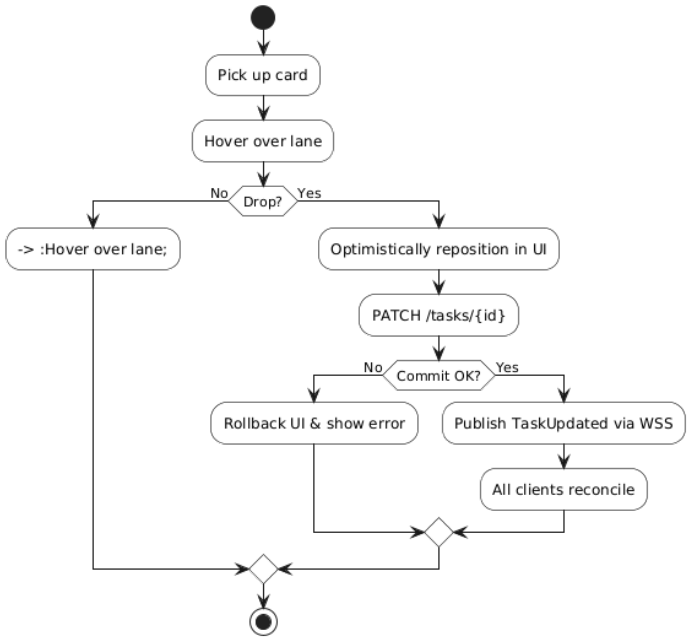
Figure 6. Create Task Workflow



Drag to Move Task Workflow

When a task card is moved on the task board, the following steps occur in sequence: drag, drop, optimistic UI update, PATCH request, commit confirmation, and real-time sync.

Figure 7. Drag to Move Task Workflow



Interaction Details & Responsibilities

- **Frontend:** Binds to user action, ensures input validation, makes REST calls, and optimistically updates the UI. Reconciliation happens when the authoritative data comes back off the server or by WebSocket update.
- **Backend:** Responsible for validation, authorization, DB transactions, and publishing events out to the WebSocket hub.
- **Database:** Holds all permanent objects (users, boards, lanes, tasks, comments, and activity logs) with indexed performance and referential integrity.
- **WebSocket Hub:** Push create/update/delete events to active clients to maintain the synchronization in real time.

Concurrency & Consistency

- **Atomic transactions:** The whole call to REST either commits or rolls back
- **Event versioning:** Each WebSocket message includes the version number and timestamp such that outdated updates cannot be accepted**.**
- **Conflict handling:** When simultaneous edits occur, the latest version wins.
- **Responsiveness:** Event propagation typically occurs within one second under normal load.

Error & Exception Path

- **Client-side:** Input validation, network recovery in case of interruption and auto-reconnection logic.
- **Server-side:** Returns explicit HTTP error codes for validation (400), authorization (403), not found (404), conflict (409), or generic errors (5xx).
- **Operational feedback:** The activity log records all successful transfers, and the metrics track latency and delivery success.

Verification Summary

It precisely captures the control flow, interaction, object states, and workflows, which are the essential elements of dynamic behavior.

- **Control Flow:** Responsive, event-driven, and synchronized.
- **Interactions:** Extensively modeled in sequence diagrams showing how each subsystem operates together.

- **State Changes:** Described in terms of state machine diagrams with object evolution.
-
- **Workflows:** Described in activity diagrams that lay out precise procedures for activities.

Final Notes

This architecture makes collaborative activity and dependable real-time updating seamless. It affords strong visibility into how the system handles user input, retains state, and provides immediate feedback across many connected clients

Boundary Conditions

Startup

- **Infrastructure**
    - **Frontend**: Vercel hosting the Next.js static build.
    - **Backend**: Single AWS App Runner service
    - **Database**: Supabase
    - **DNS/HTTPS**: Route 53
    - **Sockets**: Native FastAPI WebSocket endpoint
- **Initialization sequence**
    - DB is reachable
    - Backend container starts, runs migrations on boot
    - Frontend deploys
    - Health checks pass; app ready.

Runtime Behavior

- **WebSocket layer**
    - Single backend instance maintains in-memory connection registry and broadcasts updates to connected clients (projects/users they belong to).
- **Shutdown**
    - Container receives SIGTERM → stops accepting new connections → closes WebSockets gracefully → finishes in-flight requests → exits

Error Handling and Recovery

- **Frontend**: auto-reconnect on WebSocket close; friendly toasts on network errors.
- **Backend**: structured exception handlers; 4xx/5xx JSON envelopes; request logging.
- **Database**: retry on transient errors; safe transactions.
- **Backups**: rely on Supabase automatic backups
- **Bulk import/export**: provide endpoints or scripts that wrap pg_dump/pg_restore.
- **Migrations**: alembic upgrade head at boot.

Basic Component Types in this System

- **Use Cases**:
    - Create task
    - Update task
    - Assign task
    - Manage projects.
- **Functions**:
    - CRUD APIs
    - Authentication
    - Validation.
- **Triggers**:
    - User actions (form submit, button click)
    - System events.
- **Data Stores**:
    - PostgreSQL tables (users, projects, tasks, logs).
- **Data Flows**:
    - JSON payloads via REST
    - SQL queries between backend and DB.
- **Data Elements**:

    - User records
    - Project metadata
    - Task details.
- **Processors**:
    - FastAPI server
    - Next.js runtime.
- **Data Storage**:
    - Docker volume (pg_data)
    - Managed DB in production.
- **Data Connections**:
    - HTTP(S) between frontend & backend
    - TCP between backend & DB.
- **Actors/External Entities**:
    - End users
    - Admins
    - Vercel
    - Supabase
    - AWS

Individual Contributions

Kevin Ekart

- RF-1 – Social Feasibility
- RF-2 – Specific Requirements
- RF-3 – Boundary Conditions
- Documentation assembly and formatting

Ethan Gray

- RF-1 – Economic Feasibility
- RF-1 – Market Research
- RF-2 – System Attributes and Other Requirements
- RF-3 – Overview
- RF-3 – Subsystem Decomposition
- RF-3 – Hardware/Software Mapping

Laken Hollen

- RF-1 – Alternative Solutions
- RF-1 – Project Risks
- RF-2 – Specific Requirements
- RF-3 – Global Software Control

Huynh Le

- RF-1 – Product Feasibility
- RF-1 – Technical Feasibility
- RF-2 – Introduction
- RF-3 – Persistent Data Management
- RF-3 – Access Control and Security

Key Personnel Information

KEVIN EKART

SENIOR REACT DEVELOPER 2026 CARDINAL LANE, JEFFERSONVILLE, IN, 47130, UNITED STATES 812-697-0048

° Details °

2026 Cardinal Lane, Jeffersonville, IN, 47130, United States
812-697-0048
kevin.ekart@gmail.com

° Skills °

PHP (10+ years)

Python (7 years)

MySQL (10+ years)

MongoDB (8 years)

JavaScript (10+ years)

TypeScript (5 years)

NodeJS (4 years)

React (7 years)

React Native (3 years)

AWS (9 years)

Serverless (7 years)

Postgres (5 years)

NextJS (3 years)

Employment History

Senior React Developer at Forecastr, Louisville, KY

May 2022 — Present

- Helped transition existing platform from React to NextJS.
- Helped improve speed of the application by 60%.
- Built integration pipeline using AWS Glue to pull and aggregate data from multiple sources including QuickBooks Online, Xero, and Hubspot.

Owner and Software Engineer at Echelon Kinetic Art & Multimedia, Jeffersonville, IN

May 2012 — Present

- **Adjunct World**
  - Took over existing code base for maintenance and enhancements using PHP, MySQL, HTML, jQuery, JavaScript, and CodeIgniter.
  - Built course sign up and checkout that integrates with Stripe to process payments and keep inventory of available seats.
- **Unnamed Press**
  - Built and designed online store and admin for store using PHP, MySQL, HTML, jQuery, JavaScript, and Laravel.
  - Built cart and checkout system that integrates with Stripe to process payments.
- **Rally78**

    - Built and designed API for mobile application using PHP, Python, NodeJS, MongoDB, and Lumen.
    - Built mobile application using React Native and TypeScript.
    - Built real-time chat interface in mobile app utilizing NodeJS and integration with Pusher.
    - Served as technical lead during conversations with the USTA about partnering.
- **Core Integrated Marketing**
    - Built integration between Evosus and Sharpspring using PHP and Laravel.
    - Built integration between Salesforce and Sharpspring using PHP and Laravel.
    - Built integration that scans SFTP server for new files, parses them, and loads them into Sharpspring using PHP and Laravel.

Lead Data Architect at Capture Higher Ed, Louisville, KY

October 2020 — May 2022

- Took over the data warehouse project started by an outgoing employee.
    - Optimized scripts that export data from the production database into the data warehouse using Python, MySQL, AWS S3, AWS Glue, and AWS Athena.
    - Reduced runtime of existing scripts from 23 hours to 8 hours.
- Designed and built business intelligence reports using Python, React, and AWS Athena.
- Planned next generation of data warehouse to reduce runtime and costs.
- Operated as technical advisor for product management group, data exchange group, and operations group.

Lead Software Architect at Capture Higher Ed, Louisville, KY

May 2017 — October 2020

- Designed architecture for fourth iteration of company platform (Engage 4) moving from a monolithic style architecture to a microservices design.
- Built initial prototype of Engage 4 using PHP, HTML, MySQL, and Laravel.
- Reimagined how the company sends bulk emails to improve and maintain sending reputation.
    - Developed plan for both IP warmup and domain warmup.
    - Developed plan for avoiding spam traps.
    - Improved open rates from 12% to 28%
    - Improved click through rates from 3% to 12%.
- Operated as technical advisor within the product management group.
- Operated as top tier support for development group to help troubleshoot and resolve issues with the platform.

Manager of Product Development at Capture Higher Ed, Louisville, KY

July 2015 — May 2017

- Oversaw and managed team of 5 developers.
- Operated as a working manager spending 60% of my time coding and 40% managing the team.
- Instituted 2-week sprints using Jira to track progress.
- Designed, built, and oversaw the implementation of the third iteration of the company's platform (Engage 3) using PHP, HTML, jQuery, JavaScript, MySQL, and Laravel.
    - Built email service capable of queueing and sending millions of emails daily via Mailgun.
    - Built tool that would create screenshots of emails to share with clients for approval using NodeJS.

- Moved platform from a single instance for all clients to an instance per client for scaling.

Software Engineer at Capture Higher Ed, Louisville, KY

April 2013 — July 2015

- Designed and built the second iteration of the company's platform (Engage 2) using PHP, HTML, jQuery, JavaScript, MySQL, and Laravel.
- Moved the platform out of single PHP file to an MVC framework using a monolithic design.
- Improved efficiency of file imports reducing import times by up to 400%.
- Built mobile version of Engage 2 using Apache Cordova.
- Built prototype of company's flagship product for tracking and identifying leads on clients' sites using JavaScript, PHP, and Slim.

Software Engineer at The Learning House, Louisville, KY

April 2009 — April 2013

- Worked to design and build company's internal ERP system (Grail) to manage clients and workloads.
- Worked to design and build company's partner portal that allow clients to view and export reports.
- Built integrations from company's ERP and partner portal to clients' Moodle LMS platforms to gather course data and launch courses.
- Started building media library platform for managing images, videos, and other types of content for publication.
- Built real-time chat module with rooms for each course for Moodle to replace existing, buggy chat module using PHP, HTML, jQuery, and NodeJS.

Computer Hardware Specialist at Spalding University, Louisville, KY

March 2008 — April 2009

- Worked as primary contact for resolution to technical issues on campus.
- Responsible for installing, upgrading, and maintaining computer hardware and software for entire campus.
- Managed server network for campus library.
- Managed computer and software inventory.

Education

Bachelor of Science in Computer Science, Indiana University Southeast, New Albany, IN

January 2022 — Present

Undergraduate Certificate in Cybersecurity, Indiana University Southeast, New Albany, IN

January 2022 — May 2025

Associates in Multimedia, ITT Technical Institute, Louisville, KY

March 2006 — August 2008

Awards and Recognition

Forecastr – Speak the Truth Core Values Award

July 2023

Capture Higher Ed - MVP

August 2015

Business First - 20 People to Know - Technology & Innovation

November 2015

Capture Higher Ed - MVP

August 2013

Ethan Gray

1237 N Highway 31, Austin, Indiana (812) 216-0436 egray1333@gmail.com

EDUCATION

Austin High School Austin, Indiana

Core 40 and Academic Honors May 2022

GPA: 3.604

Ivy Tech Community College Sellersburg, Indiana

Associate of General Studies in General Studies May 2022

Indiana University Southeast New Albany, Indiana

Bachelor of Science in Computer Science Expected: May 2026

GPA: 3.329

EXPERIENCE

Math Tutor, Austin High School January – May 2020

- Greeted customers and treated them with kindness and respect, making them feel welcome in the area
- Worked with other math tutors to aid students in completing various math problems
- Maintained a clean learning environment

Dishwasher, Cracker Barrel August 2025 – Current

- Worked with other dishwashers to clean dishes and stock them in a timely manner
- Cleaned the dish room when dirty and at the end of the day
- Assisted in other areas as needed, such as taking the trash out

LAKEN HOLLEN

8487 S Riddle Rd | Leavenworth, IN 47137

812-968-5485 | Lakendawn0813@gmail.com

OBJECTIVE

Aspiring IT professional pursuing a B.S. in Computer Science with hands-on experience in tech support, digital tools, and community-focused projects.

EDUCATION

Indiana University Southeast, New Albany, IN

Bachelor of Science in Computer Science (Expected Graduation: May 2026)

- Minor: Science/Mathematics
- Dean's List: Every semester attended

Relevant Coursework:

- Programming Fundamentals
- Foundations of Digital Computing
- Introduction to Computer Software Systems
- Computer Programming II
- Intro to Operating Systems
- Computer Structures
- Computer Security
- Intro to Digital Forensics
- Data Structures

Crawford County High School, Marengo, IN

Academic Honors Diploma (May 2022)

- GPA: 3.8 (Unweighted), 4.0 (Weighted)
- National Honor Society (2020–2021)
- Honor Roll (Fall/Spring 2018–2020)
- Booster Club Member (2018–2021)

WORK HISTORY

Philanthropy/ IT Intern

Community Foundation of Crawford County | Marengo, IN | January 2025 – Present

- Provide IT support for meetings and events, including setup and troubleshooting of devices, Microsoft SharePoint, and Office 365 tools
- Developed the Crawford County Community Resource Guide, a directory of services for low-income residents seeking food, housing, healthcare, and other assistance
- Supporting outreach efforts through nonprofit site visits and survey development to better communicate the missions, needs, and impact of local organizations
- Contribute to daily operations and special projects across departments, demonstrating adaptability and a proactive mindset
- Selected as the facilitator for the Summer 2025 Code IT Academy, a tech training program hosted by CFCC in collaboration with Ivy Tech and The Mill, aimed at expanding digital skills in the community

Cook - Dietary Department

Todd Dickey Nursing and Rehabilitation – Leavenworth, IN | December 2021 – January 2025

- Prepared breakfast and lunch, often simultaneously, for up to 60 residents, ensuring meals met nutritional standards, dietary restrictions, and facility schedules.
- Efficiently organized and executed tray line service for both breakfast and lunch, helping to maintain a timely and accurate meal delivery system.
- Maintained strict sanitation standards and collaborated with kitchen and nursing staff
- Balanced workflow under tight deadlines to ensure timely service

---

PROJECTS

- **Disk Analyzer Tool** – Built a Python script to analyze disk usage and generate reports

- Board Game Website – Designed a fantasy strategy board game site with HTML/CSS and JavaScript, including dynamic forms and gameplay summaries
- Community Resource Guide – Compiled and organized local nonprofit information into a printed guide distributed to residents seeking food, housing, and healthcare assistance.

TECHNICAL SKILLS

- Programming Languages: Python, Java, C, C++, Assembly, JavaScript
- Tools: Microsoft Office, SQL (basic), Git, IntelliJ, PyCharm, Node.js, Visual Studio Code, Virtual Box, ChatGPT, Excel (basic)
- Operating Systems: Windows, Linux
- Data Analysis & Troubleshooting

SOFT SKILLS

- Communication, Time Management, Problem-Solving, Adaptability, Teamwork

HUYNH LE

CONTACT

281 - 966 - 5277
lehuynh228@gmail.com
Louisville, KY 40228

TECHNICAL SKILLS

Networking: TCP/IP, LAN/WAN, Wi-Fi basics, troubleshooting principles

Cybersecurity: Security fundamentals, firewalls, VPNs, cryptography basics

Programming & Scripting: Java, Python (basics), Bash (exposure)

Operating Systems: Windows, Linux (familiarity)

Tools & Platforms: Wireshark, VirtualBox, VMware, Packet Tracer

Analytical & Soft Skills:

- Problem-Solving & Logical Reasoning
- Data Analysis (Packet/Log review - conceptual)
- Attention to Detail
- Communication (Written & Verbal)
- Teamwork & Collaboration
- Adaptability & Eagerness to Learn

EDUCATION

Indiana University - Southeast

Expected Graduation - May 2026
B.S. Computer Science – Cyber Security

Houston Community College

2017 – 2019
A.S. Computer Science

Industrial University Of Ho Chi Minh City

2009-2013
B.A . Business Administration

Certification In Process

CompTIA Security+
(Expected Completion: December 2025)

CAREER OBJECTIVE

Motivated Computer Science student with a strong foundation in networking, cybersecurity, and technical support. Seeking an Entry-Level Computer Support Assistant position at the Help Desk to leverage my troubleshooting skills, customer service experience, and passion for technology to resolve user issues efficiently and contribute to a collaborative IT team.

PROJECTS

Virtualized Network Setup with Wireshark Traffic Analysis

Description: Configured a virtualized network using VMware/VirtualBox with multiple subnets and analyzed traffic using Wireshark to identify protocol (HTTP, DNS) and troubleshoot connectivity issues.

Skills : Virtualization, subnetting, Wireshark, packet analysis, troubleshooting.

Simulated Network Attack Detection Using Wireshark

Description: Simulated a basic ARP spoofing attack in a virtual environment and used Wireshark to detect malicious traffic, documenting findings and mitigation steps.

Skills: Wireshark, cybersecurity, virtualization, network monitoring.

File Recovery and Analysis

Description: Used tools like Autopsy to recover deleted files from a disk image and analyzed file details (metadata) to track user activity.

Skills Demonstrated: File recovery, metadata analysis, digital forensics.

Hidden Data Detection

Description: Used tools like Steghide to find and extract hidden data in image files. Documented the steps and results.

Skills: Steganography, data concealment detection, digital forensics.

Honeypot with Fake Billing Page

Description: Developed a honeypot system with a fake billing page using Python, HTML, and CSS to collect attacker credentials (ID, username, password) and log malicious activity.

Skills: Python programming, web development (HTML/CSS), cybersecurity, ethical hacking, and threat detection.

Password Strength Checker

Description: Developed a Python program to evaluate the strength of passwords based on criteria like length, use of uppercase/lowercase letters, numbers, special characters, and resistance to common patterns.

Skills: Python programming, cybersecurity, password security, algorithm design.

WORK EXPERIENCE

ASSISTANT MANAGER *Anthony Vince Nail Spa* *2020 - Present*

- Led and trained a team of 13+ employees, fostering teamwork and ensuring excellent customer service.
- Resolved client concerns to improve satisfaction and reduce complaints by 20%.
- Organized employee schedules, optimizing for peak times and seasonal demand.
- Maintained staff records (salaries, PTO, schedules) using Microsoft Office Suite.
- Managed inventory supplies using Excel to track stock levels.
- Provided day-to-day customer support and bilingual translation services (English/Vietnamese)