RF-4 - Software Detailed Design

CTRL ALT Elite

Kevin Ekart / Ethan Gray / Laken Hollen / Huynh Le

October 29, 2025

Software Detailed Design

Data Design

Goal: Build a clean, secure database for our task board app (think Trello, but custom). We're using PostgreSQL via Supabase, so we get strong data safety, built-in security, and easy scaling.

1. What Data Do We Store?

We have 7 tables that work together:

Users

- Who's using the system.
- Fields: user_id, name, email (must be unique), password_hash (securely hashed with bcrypt), and role (Admin, Project Manager, or Team Member).

Boards

- Each board is a project (e.g., "Website Redesign").
- Fields: board_id, name, and owner_id (the user who created it).

Lanes

- Columns inside a board, like "To Do", "In Progress", "Done".
- Fields: lane_id, board_id (which board it belongs to), name, and order (so we know how to sort them left-to-right).

Tasks

- The actual work items that live in lanes.
- Fields: task_id, lane_id, title, optional description, optional assignee_id (who it's assigned to), and optional due_date.

Comments

- Notes people leave on tasks.

- Fields: comment_id, task_id, author_id, content, and created_at (auto-filled).

Activity_Log

- A permanent record of what happened (for auditing).
- Fields: log_id, user_id, board_id, action (e.g., “Moved task to Done”), optional target (like a task ID), and timestamp.

Users_Boards

- A simple link table that says: “Which users can see which boards?”
- Just two columns: user_id and board_id (together, they form the primary key).

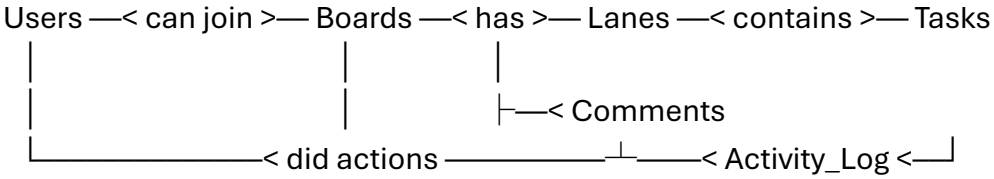
2. How Do These Tables Connect?

- A user can belong to many boards, and a board can have many users → linked by Users_Boards.
- Each board has many lanes.
- Each lane holds many tasks.
- Each task can have many comments.
- Every comment and log entry ties back to a user.
- Every log entry also ties to a board (so we know where the action happened).

When you delete a board, everything inside it—lanes, tasks, comments—gets deleted automatically.

But activity logs stay forever (we never delete history).

3. Quick Visual



4. Why This Design Works

- Safe & reliable: PostgreSQL guarantees data won’t get corrupted.
- Secure passwords: We use bcrypt—no plain-text passwords ever.

- No duplicate data: Everything is normalized (e.g., board membership is stored in one place).
- Fast performance: We'll add indexes on commonly searched fields like task_id or lane_id.
- Built for Supabase: This setup works perfectly with Supabase's row-level security (RLS), so we can lock down data by user and role.

5. Who Can See or Edit What?

You must meet two conditions to access a board:

1. You're on the board (your user_id + board_id exists in Users_Boards).
2. Your role allows the action:

Role	What You Can Do
Admin	See and edit *everything* in the system.
Project Manager	Full control over boards *you own*.
Team Member	- View any board you're invited to - Edit or comment on tasks *assigned to you* - Move tasks around within the board

Every query in the app will check both of these rules—automatically.

Architecture Design

Purpose

This section explains how information moves through the system and how that flow is organized into different parts of the program. The goal of this design is to make the system easy to understand, maintain, and update. We use a method called transformation mapping to show clear boundaries between how data enters the system, how it is processed, and how results are sent back out. A Data Flow Diagram (DFD) is used to show how the system handles input, processing, and output.

How Information Flows in the System

The Task Manager System receives information from users and then provides information back to them. The two main types of information flow are:

1. **Incoming Information** – This is what the user sends to the system.
   Examples: logging in, creating a task, updating a task, or joining a project board.
   This information enters the system through HTTP requests or WebSocket messages.
2. **Outgoing Information** – This is what the system sends back to the user.
   Examples: responses to actions (like "Task Created") or real-time updates sent to all users working on the same project.
   This leaves the system as HTTP responses or WebSocket messages.

To keep things clear and organized, the system is divided into three main modules that each handle a different part of the flow.

Transformation Mapping

Using transformation mapping, the information flow is turned into a program structure with three main modules, each with its own job:

Module	What It Handles	Example Responsibilities
Input Module	Receives information from the user and checks if it is valid	Login request, task creation request, user authentication
Processing Module	Applies the rules of the system and updates data	Task rules, updating a task, saving data to the database
Output Module	Sends results back to the user or to other users in real time	Sending success messages, sending errors, broadcasting task updates

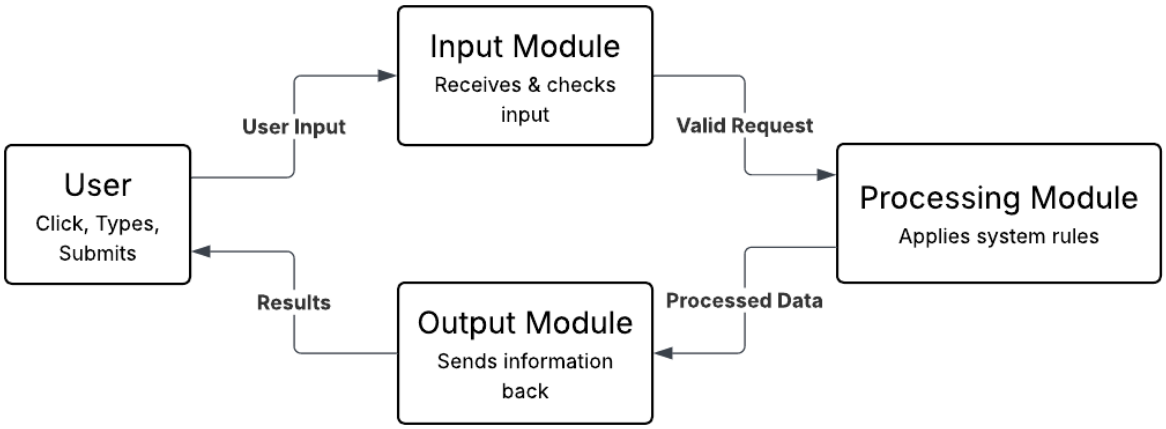
Why this is helpful:

- We can change the input (for example, add a mobile app) without changing how the system works inside.
- Business rules stay in one place so they are easy to update.

- Sending information back to users can be upgraded (for example, adding push notifications) without changing the logic that processes tasks.

Data Flow Diagram

This diagram shows how information moves through the three parts of the system:



How Each Module Handles Its Part

Stage	What Happens
Input (Control Input)	The system receives the request, checks login, and makes sure the information is correct.
Processing	The system applies the rules, makes decisions, and stores or updates information.
Output	The system sends a response, such as success or error messages, and sends real-time updates to other users.

Summary

This architecture design clearly separates how the system receives data, processes it, and sends results back. By using the transformation mapping method, the system becomes easier for new developers to understand and work on. It also allows future changes to be made without rewriting the whole system, which is helpful for long-term maintenance and improvements.

Interface Design

Internal Interfaces

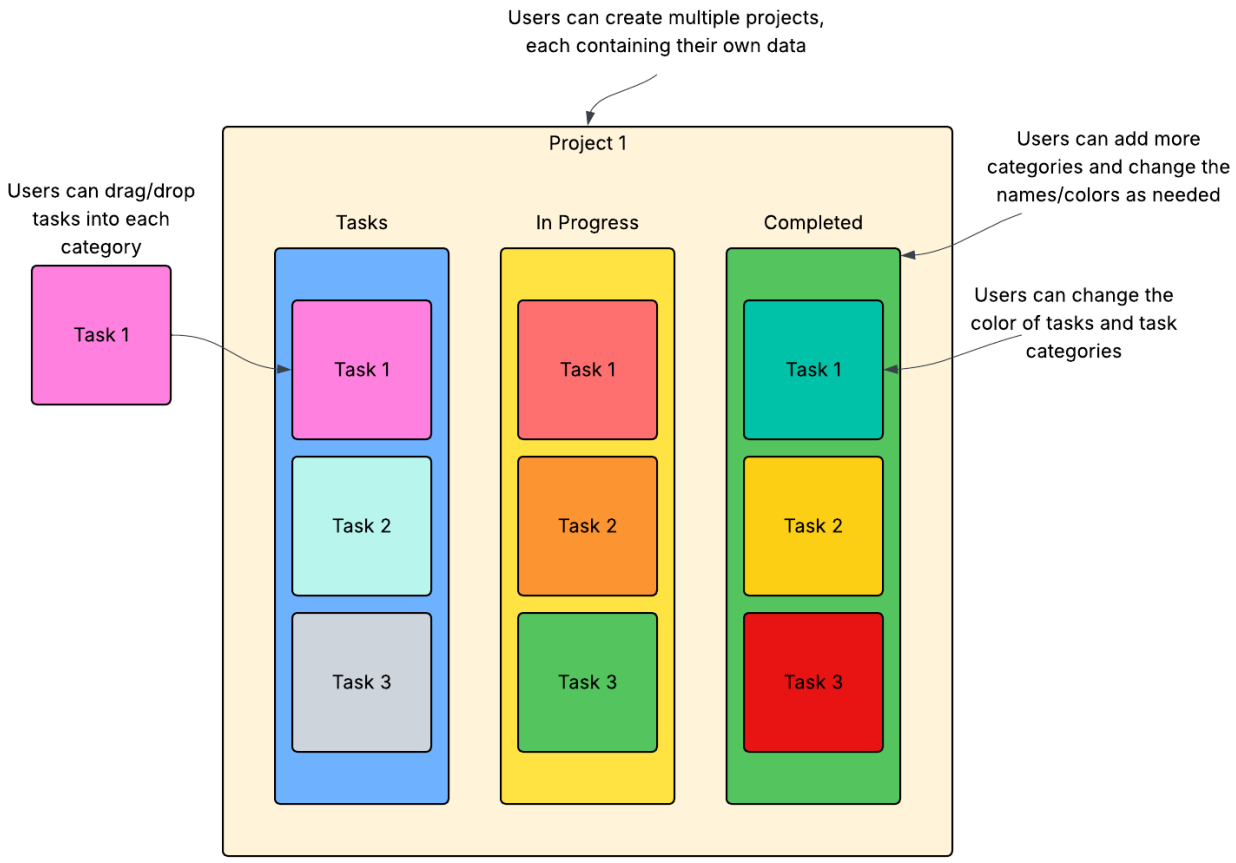
- **Database Console** – Administrators will use the console to view audit logs, make changes to the contents of the database, and start/stop the system as needed.
- **Text Editor** – Administrators can use any text editing software of their choice to make changes to the code running the system. A system restart will need to occur for any changes to take effect.
- **Internet Connection** – The system will need internet access to function and allow administrators to interact with it.

External Interfaces

- **Application Server** – This server will manage the connection to the client. It will handle any data transmitted between the client and the database, such as new database submissions or authentication requests.
- **Database Server** – Project/task information and authentication attempts require communication between the client and the database server. Constant communication will need to persist as long as the client is connected, so that the system is ensured to receive all the data the client submits.
- **Internet Connection** – Any communication between the client and the system will occur via an internet connection, wired or wireless.

Human Interface

- **Peripherals** – monitor, mouse, keyboard
- **GUI** – The primary way the user interacts with the software is through the graphical user interface. Some of the ways they use the GUI include authentication and the adding, removing, or editing of tasks.
- Below is an example of what the project GUI *may* end up looking like. However, at this time, no specific plans exist for what it will look like.



Procedural Design

Purpose

The procedural design describes the internal working of each aspect of the digital task management board. It helps to turn the architecture and the needs of the system into logical program components that can be implemented by the developer. The system follows structured programming and is described below:

- **User Interface Layer** (Nextjs + React components)
- **Application Logic Layer** (Flask API controllers)
- **Data Access Layer** (PostgreSQL via SQLAlchemy ORM)

This design facilitates ease of understanding and testing capabilities that will allow a novice programming team to write the code that is based on the design.

Components and Processes

1. Authentication Component

Purpose: Check the credentials of users and restrict access to boards

Input: Email, password

Output: Authenticated session token or error message

Process:

- Get credentials from the login form
- Validation of User Record in database
- Generate session token (JWT)
- Return token to the client for subsequent operations

Design Constraints: HTTPS protocol and bcrypt hashing algorithm used.

Pseudo-Code:

```
function login(email, password):
  user = dbfind_user(email)
  if not user:
    return error("Invalid credentials")
  if bcryptverify(password, userpassword_hash):
    token = create_session_token(userid)
    return token
  else:
    return error("Invalid credentials")
```

2. Task Management Component

Purpose: Control task creation, updating, moving, and deletion

Input: Task title, description, lane_id, assignee_id, due_date

Output: Task list updated or confirmation message

Process:

- Accept a JSON payload via the REST /tasks endpoint
- Validate required fields
- Perform a database operation (INSERT/UPDATE/DELETE)
- Send update via WebSocket

Design Constraints: Each operation must commit atomically; changes must be sent to each client connected.

Pseudo-Code:

```
function update_task(task_id, changes):
  task = dbget_task(task_id)
  if not task:
    return error("Task not found")
  taskupdate(changes)
  dbcommit()
  broadcast("task_updated", task)
```

3. Board Management Component

Purpose: Create, modify, or delete project boards and project lanes.

Input: Board name, owner_id, lane structure

Output: Board ID and confirmation

Process:

- Create board entry
- Setup initial lanes (“To Do”, “In Progress”, “Done”)
- Assign ownership and permissions

Pseudo-Code:

```
function create_board(name, owner_id):
  board = Board(name=name, owner_id=owner_id)
  dbadd(board)
  dbcommit()
  for lane_name in ["To Do", "In Progress", "Done"]:
    create_lane(boardid, lane_name)
  return board
```

4. Comment Component

Purpose: Enable users to add notes and updates on tasks

Input: Task ID, author_id, comment text

Output: Verification and transmission of new comment

Process:

- Accept POST request /comments
- Insert new comment into database
- Broadcast data to subscribed clients

Design Constraints: Need to sanitize input against injection/XSS attacks

5. Activity Log Component

Purpose: Record actions of each user for accountability purposes

Input: user_id, action, target_type, target_id

Output: Log entry stored in database

Process:

- Each CRUD operation calls log_activity()
- Activity table records timestamped events

Pseudo-Code:

```
function log_activity(user_id, action, target_type, target_id):
    log = Activity(user_id, action, target_type, target_id, timestamp=now())
    dbadd(log)
    dbcommit()
```

6. Real-Time Update Component

Purpose: Coordinate all connected clients; keep synchronized

Input: Change event from a backend process

Output: WebSocket message

Process:

- WebSocket broadcast process when DB commits
- Clients that receive the broadcast can re-render the UI state.
- Design Constraints: Latency < 1 second and disregard messages based on timestamp.

Performance & Design Constraints

- Every request must be executed in < 200ms, under normal conditions
- WebSocket broadcasts must be communicated to all connected clients within 1 second
- The database must maintain referential integrity (PostgreSQL ACID compliant)
- Caching will be implemented for frequent queries, and pagination may be implemented for larger boards.

Hardware, Database, and Language

Layer	Technology	Purpose
Frontend	**Nextjs + React (TypeScript)**	Client UI, task board rendering

Layer	Technology	Purpose
Backend	**Flask (Python)**	Business logic and WebSocket updates
Database	**PostgreSQL**	Persistent data storage
Platform	**Linux (Ubuntu)** hosted via AWS / Vercel	Deployment and runtime environment

Summaries of Major Routines

Routine	Description
login()	Confirms a user's identity, issues a security token
create_board()	Creates a new project board with the default lanes
create_task()	Adds a task to a specified lane
update_task()	Updates an existing task and broadcasts the update
delete_task()	Deletes a task and all associated comments
add_comment()	Adds a comment to a task and notifies all users viewing the task
log_activity()	Logs an audit trail for any user action

ER-Level Data Model

Entities: Users, Boards, Lanes, Tasks, Comments, Activity Logs, User_Board

Key Relationships:

- Users - Boards (many-to-many via User_Board)
- Boards - Lanes (1-to-many)
- Lanes - Tasks (1-to-many)
- Tasks - Comments / ActivityLogs (1-to-many)

Individual Contributions

Kevin Ekart

- RF-1 – Social Feasibility
- RF-2 – Specific Requirements
- RF-3 – Boundary Conditions
- RF-4 – Architecture Design
- 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
- RF-4 – Interface Design

Laken Hollen

- RF-1 – Alternative Solutions
- RF-1 – Project Risks
- RF-2 – Specific Requirements
- RF-3 – Global Software Control
- RF-4 – Procedural Design

Huynh Le

- RF-1 – Product Feasibility
- RF-1 – Technical Feasibility
- RF-2 – Introduction
- RF-3 – Persistent Data Management
- RF-3 – Access Control and Security
- RF-4 – Data Design

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)