Web & Mobile Developer

Ju-cheol Park

Web applications and Android integration, with selected work in settlement, logistics, and stock administration.

Experience

Triphos Co., Ltd. / 2026.02 ~ Present

Scope

Built web interfaces and server APIs, and integrated Android app features

jukrap628@gmail.com github.com/jukrap www.linkedin.com/in/jukrap/ jukrap.vercel.app/en

Selected work

01 Settlement ERP Platform

Built the ERP web app, key server APIs and data persistence, with shared editing tables, batch Excel imports and administrator work logs.

02 Logistics Operations Web and Printing App

Built search, reservation and Excel-import screens, API integration and Android label printing. Reduced the initial JavaScript entry file by about 74% (build output size).

03 Stock Operations Admin Web

Built the complete React frontend using MSW mock APIs, with shared tables and query-state handling for account, stock and order screens.

Next, I will introduce daily information and privacy apps, an internal documentation tool, further company work, and personal and team projects.

Company work 01

Settlement ERP Platform

A business system for vehicle and driver records, transport fees, billing, and settlement. Built the ERP web app and key server APIs, and extended the existing user-facing web and mobile apps.

Period	Scope	Source contribution
2026.07 ~ 2026.09	Full-stack development	3,079 files / 424,000 lines

Technologies React, TypeScript, Tailwind CSS, Vite, React Router, Zustand, Spring Boot, MyBatis, MariaDB, Vitest

From data entry to batch saves

Editing tables that retain input

Built keyboard cell navigation, selection dialogs, and per-row save feedback, retaining edits in rows that fail to save.

From Excel preview to persistence

Highlighted invalid Excel rows and cells for correction and revalidation. Rolled back failed batch imports and returned existing results for repeated requests.

Record changes and settlement history

Separated current assignments from historical transactions to retain finalized settlement amounts. Vehicle save feedback identifies newly created records, operating-status changes, and affected insurance or utility records.

Logs showing actions and errors

The initial logs captured many details, but technical fields made it difficult to follow what had happened. I surfaced affected records, changes, and input-error reasons, and removed unnecessary technical fields.

Company work 02 / Web

Logistics Operations Web

A logistics web app for search, reservations, Excel imports, and label printing. Built the complete frontend with React and TypeScript, including server API integration and PC label printing.

Period	Scope	Source contribution
2026.04 ~ 2026.06	Complete frontend development	403 files / 63,000 lines

Technologies React, TypeScript, Vite, React Router, TanStack Query, Zustand, XLSX, Vitest

Business interfaces and printing

Shared interfaces and API integration

Built shared tables, forms, and modals, connecting server APIs to workflows from search and reservations through Excel previews and print requests.

Long addresses on printed labels

Wrapped PC label addresses by measured text width and fitted the remaining address into the final line.

Load only the code needed on entry

The initial bundle included multiple pages and heavy Excel code that users did not need on their first screen.

Identified code that was unnecessary on entry and loaded pages and Excel libraries only when needed.

Bundle size change

Initial JavaScript entry

2,405.50 → 616.59 kB

About 74% smaller on the same build basis

Gzip-compressed size

815.10 → 204.38 kB

About 75% smaller for the same initial entry

Scope of code splitting

Also evaluated splitting shared authentication and API initialization code. This reduced the initial file size further, but complicated authentication initialization and loading on the first interaction, so I did not adopt it. Applied lazy loading only to pages and Excel libraries.

Company work 02 / Android

Logistics Label Printing App

Connected web label-print requests to an Android app and Bluetooth printers.

Period	Scope	Source contribution
2026.04 ~ 2026.06	Android app development	81 files / 7,000 lines

Technologies Expo, React Native, Expo Router, WebView, Android native module, TypeScript, Bluetooth

Context

Web print data needed to reach a native device SDK while the interface handled permissions, connection, and cancellation.

Implementation

Label-printing path

01	02	03
WebView	**Android**	**Bluetooth**
Label data and print request	Bridge, permissions, device connection	Printer SDK and label output

From web requests to hardware

Passed print data through a native bridge and implemented device selection, connection, and print calls as a single flow.

Permissions and cancellation

Traced SDK discovery calls on Android 16/API 36 and corrected cancellation and BLUETOOTH_SCAN, BLUETOOTH_CONNECT permission handling.

Outcome

Verified Bluetooth connection and physical label output on an Android device.

Company work 03

Stock Operations Admin Web

Built the complete React frontend for a stock administration web app using MSW mock APIs.

Period	Scope	Source contribution
2026.02 ~ 2026.03	Complete frontend development	182 files / 18,000 lines

Technologies React, TypeScript, Vite, React Router, TanStack Query, Zustand, MSW

Shared tables, filters, and modals

Built shared list, search, detail and editing components for account, stock and order screens. Added column pinning, resizing and reordering to tables, and implemented queries and input interactions with MSW mock APIs.

Applying order-history filters

Kept account and stock selections separate from the conditions applied to the list. Editing a filter retained the existing results; pressing Search applied the submitted conditions to the TanStack Query key.

Repeat searches and load more

Search again with the same filters

Reset the existing query to reload from the first page.

Guard load-more requests

Skipped additional requests when no next page remained or a request was already in progress.

Company work 04

Weather in a Daily Information App

Added public-data weather features to an existing Android app, implementing the WebView interface, server-side API integration, and caching.

Period
2026.06 ~ 2026.07

Scope
Frontend and backend feature development

Technologies Spring MVC, JSP, jQuery, Java, Android

Weather data across separate APIs

Weather, air quality, and alerts had different regional coverage and update schedules. Fetching everything on every visit repeated requests for the same area and could hold up available information while a slower response arrived.

Caching by region and data type

Reuse data for requested regions

Cached responses by region and data type, with expiry periods matched to each type.

Share in-flight requests

Reused an in-flight result for the same cache key rather than issuing another external API call.

Show core information first

Displayed current conditions and the core forecast first, then filled in air quality and alerts as responses arrived. Delays or failures in supplementary requests did not hide weather data already received.

Handling stale information

When an external request failed, allowed stale values could remain visible. An expired “no alerts” response was marked unavailable rather than treated as current. Previously retrieved active alerts were labelled as based on the last retrieval.

Company work 05

Privacy Check App

Added email-breach lookup and deepfake image checks to an existing Android app. Connected inspection screens, server calls to external APIs, and Android image selection from input through results.

Period
2026.06

Scope
Frontend and backend feature development

Technologies Android, Cordova, Spring MVC, jQuery, Java

Turn external inspection APIs into app features

Implemented server calls to external APIs for submitted emails and images, translating responses and errors for the interface. API credentials remained on the server; the app received the results and status needed for display.

From image selection to inspection

Input validation and preview

Validated file type, size, and resolution, and displayed a preview of the selected image.

Android file selection

Connected image and file requests from the WebView to native selection results.

When the user selects another image

A previous file read can finish after the user selects another image. Assigned a token to each read and checked it on completion so stale results would not overwrite the new preview.

Company work 06

AI-assisted Project Documentation Web App

Built an internal web tool for drafting and reviewing documents from repositories and project kickoff material, covering source collection, table editing, and export.

Period
2026.04

Scope
Frontend and backend development

Technologies Node.js, TypeScript, React, AI API, xlsx, Vitest

Review source material before drafting

Split the interface into review stages for proposed requirements and open questions before drafting final documents. Enabled functional specifications after the requirements document was ready, and screen specifications after their prerequisite documents were ready.

Review tables and revise selected content

Edit document sheets

Displayed requirements, functions, and screens as editable sheets and exported documents from the edited data.

Targeted AI revisions

Sent the selected sheet and cells with surrounding content to distinguish revision targets from context to preserve.

Handle partial generation failures

Tracked generation status per document so completed documents remained available for review after another failed. Supported regenerating and exporting selected documents without restarting the entire set.

Company work

Further company work

Feature additions to existing services and updates for Android versions and field use.

07 Member and Inquiry Management Web and App

Period
2026.09

Platform
Web / Android

Implementation

Extended an existing solution with member search, editing, merging, inquiries, and attachments. Integrated file selection and image previews in the Android app.

Technologies Spring MVC, Java, MyBatis, Vue 2, MariaDB, Android, Kotlin

08 Parcel Delivery App

Period
2026.08

Platform
Web / Android

Implementation

Updated map-marker selection, current location and return-to-map behavior. Implemented offline delivery-completion saves, request recovery after app restarts and retries after reconnection.

Technologies React, TypeScript, React Native, Expo, Spring Boot, SQLite

09 Chart Preview Page

Period
2026.03 ~ 2026.04

Platform
Web

Implementation

Built chart settings panels, data connections, and previews as part of a website that was under development.

Technologies React, TypeScript, Vite, Chart.js, WebGL/GLSL (OpenGL-style), OGL (WebGL library), MSW, Vitest

10 Logistics Receiving and Shipping PDA App

Period
2026.03 ~ 2026.04

Platform
Android

Implementation

Updated an existing PDA app for newer Android versions and added receiving screens, quantity entry, and QR/barcode scanning. Revised the app update download and installation flow.

Technologies Android Java, Gradle/AGP, Scanner SDK

11 Legacy Shipping and Inventory App

Period
2026.03

Platform
Android

Implementation

Updated the build environment of a legacy Android app with a WebView interface, adapting permissions, file access, and back navigation to OS changes. Verified behavior on devices running several older Android versions.

Technologies Android Java, Gradle/AGP, RxJava, FileProvider

Project 01

C. Donghae

Built a map-based web service for Donghae Line passengers to find live transport and nearby information.

Period
2024.10.04 ~ 2024.10.06

Role
Frontend developer / team of three

Technologies TypeScript, React, Next.js, Tailwind CSS, Google Maps, Storybook, AWS, GitHub Actions, Docker

Hackathon scope

The team had 72 hours to combine train, weather, and local-place data in one map with mobile interactions.

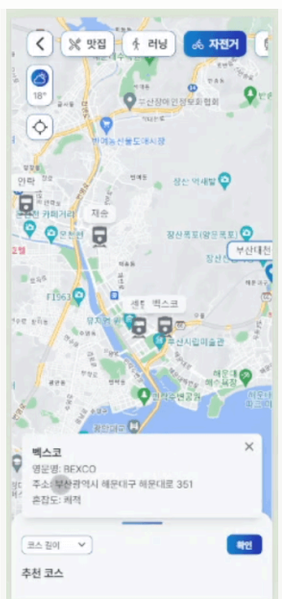
Implementation

Maps and API integration

Worked with two backend developers using Swagger documentation to connect APIs and build the main Google Maps screens.

Mobile map interaction

Built a bottom sheet over the map that expands or collapses according to drag distance and velocity.



Award

Won the Busan Technopark President Award at DIVE 2024, placing third in the challenge track.

Project 02

ShareBBy

Developed Android support and community features for an app that helps people share hobbies and find companions.

Period
2024.04 ~ 2024.06

Role
React Native app development and Firebase integration

Technologies JavaScript, React Native, Firebase, Faster Image

Android support and image updates

Screens initially built for iOS needed Android support, and stale image caches prevented updated post images from appearing.

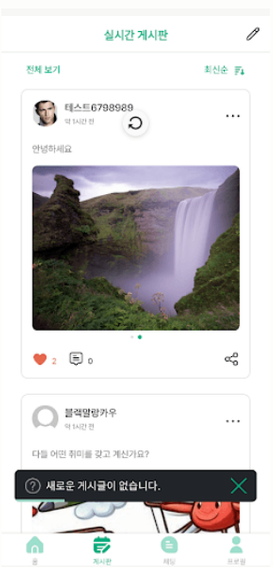
Implementation

Community features

Built post and comment editing, location filters, sorting, pull-to-refresh, and infinite scrolling.

Data and image handling

Documented Firebase relationships in an ERD, reproduced the caching problem, and adopted a maintained replacement library.



Release

Delivered Android screens and community features. The team released the app on the App Store in 2024.

GitHub
github.com/jukrap/archived-rn-ShareBBy

Project 03

AI Agent Playbook

Built a harness that helps AI coding tools carry project rules and work history across sessions. Added selected skill installation, migration and recovery while preserving existing files.

Period
2026.06 ~ Present

Role
Solo developer

Technologies JavaScript, Node.js, MCP SDK, Zod, GitHub Actions

Context and user-file preservation

Working across repositories required shared records to remain distinct from local history. Updating skills also had to preserve user edits and avoid losing a working setup after a failed installation.

Implementation

Shared records and read-only MCP

Implemented CLI and MCP access to registered repositories' records. Paginated long responses and rejected continuation after the underlying source changed.

Preserving edits and recovering installations

Checked ownership and file hashes, keeping backups and recovery records. Profile migration requires every selected replacement to be valid and rechecks them before each removal.

Release and verification

Published on npm and GitHub, with installation and usage documented in English and Korean. Tested installation conflicts, partial failures, recovery, query scope and continuation, and configured Windows/Ubuntu CI.

GitHub
github.com/jukrap/ai-agent-playbook

npm
www.npmjs.com/package/ai-agent-playbook

Project 04

Itzip

Built blog and Markdown editing screens in a 15-person team, adding tests and error monitoring to the development workflow.

Period

2024.07 ~ 2025.06

Role

Frontend team lead / DevOps

Technologies TypeScript, React, Next.js, Tailwind CSS, React Spring, Jotai, Jest, Prisma, Storybook, Postman, Sentry, AWS, Jenkins, Docker

Editing and team development

The team needed a shared way to inspect component behavior and errors after deployment while building the editor.

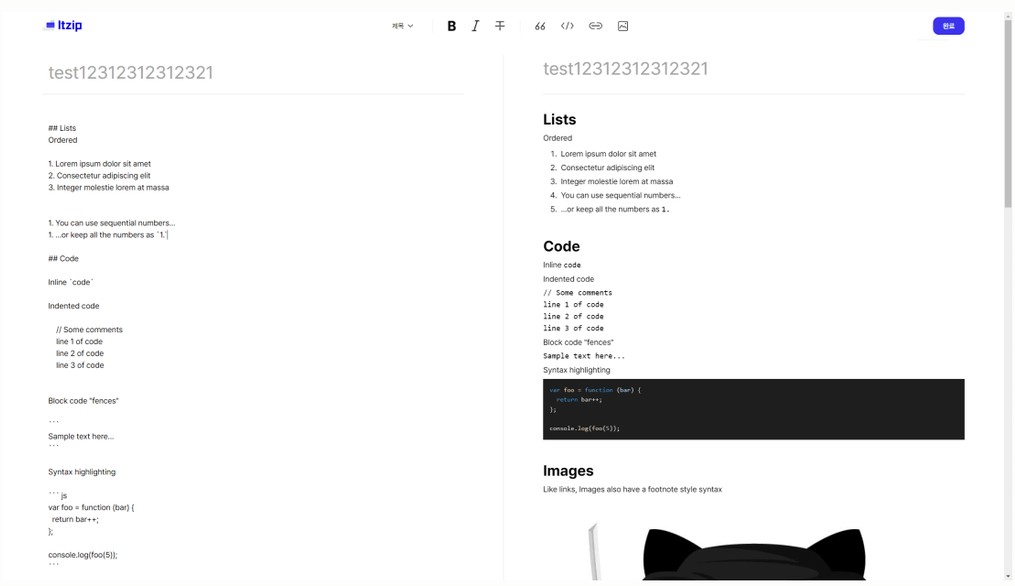
Implementation

Markdown editing and previews

Implemented live previews and project-specific Markdown syntax, splitting heavy components with dynamic imports.

Tests and error monitoring

Tested key behavior with Jest, documented components in Storybook, and configured Sentry for post-deployment errors.



Markdown editing and preview

Team adoption

Delivered blog and editor screens with unit tests, component documentation, and error monitoring.

GitHub

github.com/ITZipProject/itzip_front

Project 05

Posture Teacher

An Android app that classifies posture from body landmarks in camera frames and records exercise time.

Period
2022.07 ~ 2023.05

Role
Android developer / team of two

Technologies Java, Android, Jetpack, MediaPipe, SQLite

Camera analysis and interaction

Continuous frame analysis needed to remain separate from screen input, and MediaPipe had to be built for the Android project.

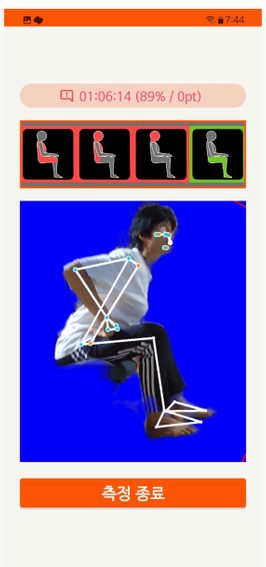
Implementation

MediaPipe integration and posture detection

Built a MediaPipe AAR on Ubuntu and used landmark angles and lengths to classify posture and measure hold times.

Separate frame analysis

Moved analysis to a separate thread so camera processing and screen input did not block each other.



Posture analysis and timing records

Measured result

During the project, frame-processing FPS was 5–10 times that of the OpenCV implementation, measured as frame-processing throughput.

GitHub
github.com/jukrap/archived-Posture-Teacher

Profile

Skills, education, and awards

Skills

Languages

TypeScript, JavaScript, Java

Mobile / hybrid

React Native, Expo, Android, Gradle

Tools & Data

GitHub Actions, Jenkins, Docker, MariaDB, SQLite

Web

React, Next.js, Vite, TanStack Query, Zustand

Testing / monitoring

Vitest, MSW, Storybook, Sentry

Education & activities

Gyeongsang National University / Computer Science

Graduated 2023.02

GPA 3.64 / 4.5

Programmers Devcourse

2023.12 ~ 2024.05

Cloud Application Engineering, React and React Native

Programmers Devcourse / Assistant Mentor

2024.05 ~ 2024.09

Shared development resources and supported daily scrums and project reviews.

Awards

DIVE 2024 Global Data Hackathon

2024.10

Busan Technopark President Award / third in the challenge track

Gyeongnam Software Competition

2021.10

ESD HotDeal / Top Excellence Award

More

About
jukrap.vercel.app/en/about

Work
jukrap.vercel.app/en/work

Side Projects
jukrap.vercel.app/en/projects