At **Yellow.ai**, we are automating the world's customer support. To move fast externally, we must first automate ourselves internally.

We are looking for a **Builder**—an entrepreneurial engineer who can code (or prompt code), but cares more about solving real business problems than debating abstract architectures. You will act as a **Forward-Deployed Engineer** for internal teams, identifying bottlenecks across **Sales, Customer Success, and Implementation**, and building AI-powered tools and portals to solve them.

This is **not** a "fix bugs from a backlog" internship. You will own the **entire lifecycle** of internal products—from interviewing a Sales VP about pain points to shipping a production-ready Next.js application that solves them.

What You Will Build

The Central Nervous System

- Design and develop a unified internal portal using **AI prompts, React, and Next.js**
- Consolidate fragmented workflows and track team efficiency

Automate the "Boring" Stuff

- Shadow CS and Solutions Engineering teams
- Identify manual, repetitive tasks
- Build **Python scripts or mini-apps** to automate them

Solve GTM Problems with AI

- Leverage **LLMs** to build intelligent workflows
  Examples:
    - Auto-fill Security Questionnaires for the Sales team
    - Build an agent that scans customer tickets to predict churn risks

Run Growth & Adoption Experiments

- Measure real business impact
- Example: *Did this tool reduce Implementation setup time by 20%?*

Who You Are

The Entrepreneurial Engineer

- CS / Engineering student who prefers building over theory
- Wants to work close to **revenue and GTM teams**
- Motivated by measurable business impact

A Shipper

- Prioritizes shipping working solutions over “perfect” code
- Comfortable with fast iterations and ambiguity

Full-Stack Capable

- Can independently build frontend (**React**) and backend (**Node.js / Python**)

Curious & Bold

- Not afraid to ask *“Why do we do it this way?”* to senior leaders in Sales or CS

Technical Requirements

- **Frontend:** Prompt-based coding with **React.js / Next.js** (mandatory)
- **Backend & Automation:**
    - Node.js for backend services
    - Python for automation scripts
- **AI / LLMs:**
    - Experience with **OpenAI / Anthropic APIs**
    - Strong prompt engineering skills
    - Ability to build useful LLM wrappers
- **Databases:**
    - Ability to design and spin up **SQL or NoSQL** databases

ASSIGNMENT

Title: Banking Agent

1. Objective

Design a System Prompt and the Interaction Logic for a Gen AI Banking Agent. The agent must authenticate users, handle multi-step API workflows, optimize token usage for intermediate steps, and render a final Dynamic Rich Media Card.

2. The Scenario

You are building an agent for "Yellow Bank." A user initiates a conversation to view their loan details.

The Happy Path:

1. **Intent Recognition:** User asks for loan details (Example, “I want to check my loan details”, “Show loan details”)
2. **Data Collection:** The Agent requests:
    - Registered Phone Number.
    - Date of Birth (DOB).
    - **User Action:** User shares their Phone Number and DOB.
3. **OTP Trigger & Verification:**
    - The agent calls the triggerOTP workflow after collecting the Phone Number and DOB.
    - **API Call:** The workflow internally calls a mock OTP API. This API does not send an actual OTP to the user's phone number; instead, it generates one of the following predetermined values for the OTP: 1234, 5678, 7889, or 1209.
    - **Workflow Output:** The OTP’s value is returned to the Agent.
    - **Agent Response:** The agent asks the user to provide the received OTP.
    - **User Action:** User shares the OTP.
    - **Agent Action:** The agent verifies the OTP successfully.
4. **Loan Account ID Retrieval (Workflow A):**
    - The agent triggers the getLoanAccounts Dynamic Rich Media (DRM).
    - **API Call:** The DRM linked workflow internally calls a mock API that returns a list of Loan Account IDs associated with the user. (Each Loan Account ID will have following attributes: Loan Account ID, Type of Loan, Tenure)
    - **DRM Output:** Display the Loan Account IDs as interactive Cards. The loan account ID must be included in the "message" value of the "select" button for each card.
    - **User Action:** User selects one Account ID.

5. **Details Retrieval (Workflow B):**
    - The agent triggers the loanDetails Dynamic Rich Media (DRM) and passes the selected Account ID as the workflow input.
    - **API Call:** The Dynamic Rich Media internally calls a mock API that returns raw technical data (tenure, interest_rate, principal_pending, interest_pending, nominee).
    - **DRM Output:** Display the customer's Loan Account details using quick replies. Include a button labeled "Rate our chat" that, when clicked, directs the user to a Customer Satisfaction (CSAT) survey agent.

The Edge Cases and Additional Agent:

- At any point (specifically after providing the phone number or seeing the loan list), if the user says, "Wait, that's my old number," or "I want to check for a different number," the agent must:
    a. Clear the previous authentication slots (Phone/DOB).
    b. Retain the intent (Viewing Loan Details).
    c. Restart the collection flow immediately without hallucinating and skipping the mandatory steps.
- Handling failure scenarios (e.g., API failures, incorrect OTP submission, invalid phone number).
- Create a CSAT agent to collect ratings (Good, Average, Bad) and feedback from the user.

3. **Technical Constraint**

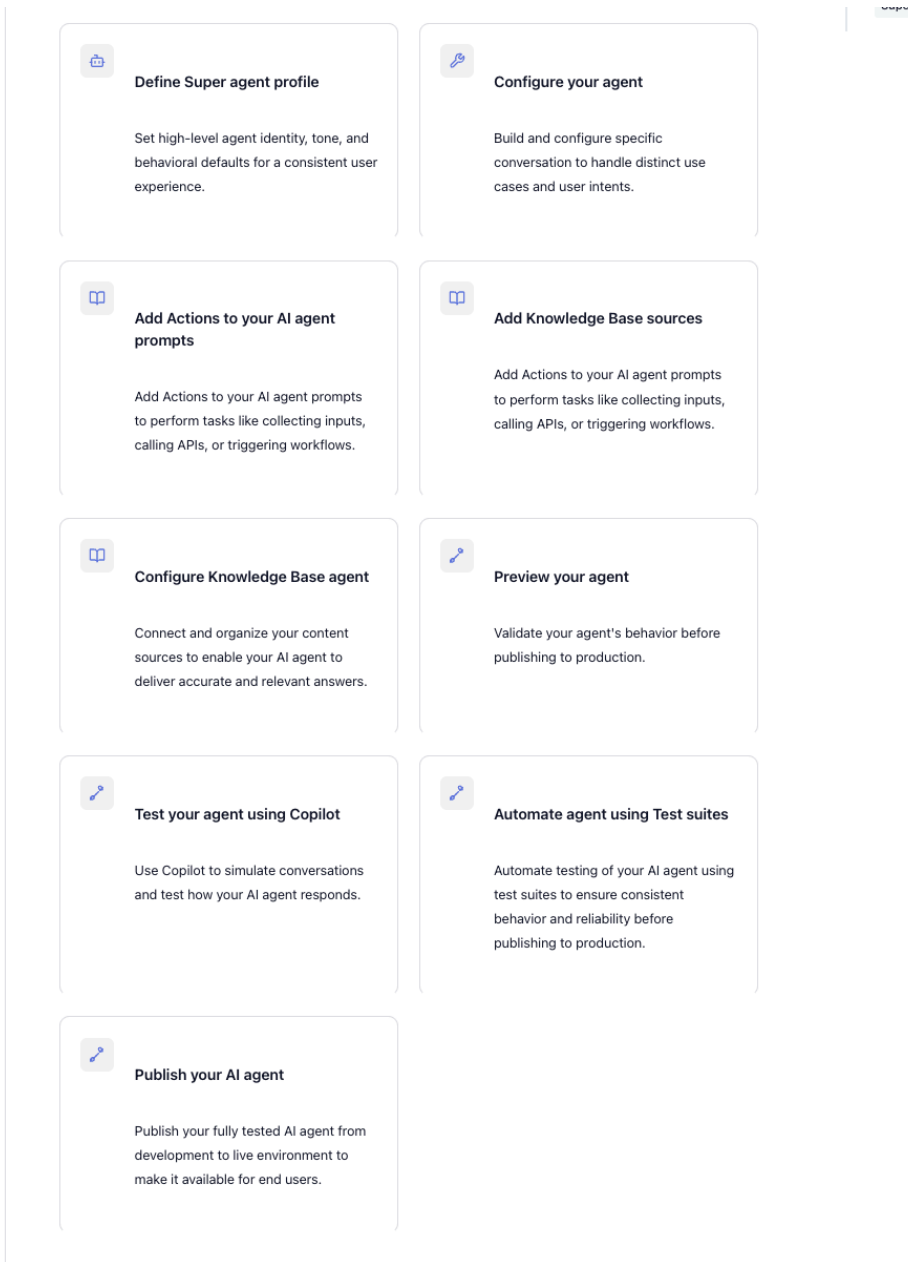
- **Token Optimization (Critical)**
  The getLoanAccounts workflow should contain an API that returns a massive JSON with 15+ fields within the objects containing loan account IDs (internal bank codes, audit date, etc.).

  Requirement: You must define a "**Middle-man Instruction**" or "**Projection**" method. that ensures the LLM only receives needed data to save tokens and prevent hallucinations.

- **Bot Language**
  The Agent should only be allowed to converse in the **English** language. If the user attempts to engage the agent in any other language, it must politely decline and state that it is restricted to operating in English only.

4. **Resources**

- Refer to https://docs.yellow.ai/ for detailed signup steps. This website also serves as a comprehensive guide to understanding YAI's platform capabilities and bot features.
- Access the following link and navigate to the bottom of the page to locate the tabs mentioned below:



Define Super agent profile

Set high-level agent identity, tone, and behavioral defaults for a consistent user experience.

Configure your agent

Build and configure specific conversation to handle distinct use cases and user intents.

Add Actions to your AI agent prompts

Add Actions to your AI agent prompts to perform tasks like collecting inputs, calling APIs, or triggering workflows.

Add Knowledge Base sources

Add Actions to your AI agent prompts to perform tasks like collecting inputs, calling APIs, or triggering workflows.

Configure Knowledge Base agent

Connect and organize your content sources to enable your AI agent to deliver accurate and relevant answers.

Preview your agent

Validate your agent's behavior before publishing to production.

Test your agent using Copilot

Use Copilot to simulate conversations and test how your AI agent responds.

Automate agent using Test suites

Automate testing of your AI agent using test suites to ensure consistent behavior and reliability before publishing to production.

Publish your AI agent

Publish your fully tested AI agent from development to live environment to make it available for end users.

Review each resource to learn the process of constructing AI Agents.

- Refer to https://beeceptor.com for deploying mock APIs in a few seconds without any dependencies. This tool allows for instant API mocking and testing, eliminating the need for complex backend setups.

5. **Assignment Submission**

Submission Instructions for Yellow Bank Agent Assignment:

1. **Share Bot Access:** Grant access to your created bot for review to the following individuals (refer to the yellow.ai documentation for detailed access sharing steps):
    - Manasvi Sharma
    - Kushagra Shrivastava
2. **Rename Bot:** Change the name of your bot to the following format:
    - <Your Name> Yellow Bank Agent
    - *Example:* Manasvi Sharma Yellow Bank Agent