Eco System Partners

StartupTN · iTNT TAMILNADU TECHNOLOGY HUB · Institute of Aeronautics Astronautics & Aviation · vidhai

HACK∞FINITY

Title Sponsor

DiGiMAXX

Event Partners

Startup DOT IN · StepUP Incubation Cell, SIF SIMATS ENGINEERING · AIC SIMATS

Problem Statement Sponsors

Valar DIGITAL COMMERCE · cmd runner

Technical Partners

theprevc. · PULLINAM AEROSPACE TECHNOLOGIES (P) LTD Softer, lighter and greener · DiffuseAi Innovate if it Impacts · Transista Technologies

Qbitia · Google Developer Groups Saveetha Institute of Medical & Technical Sciences Thandalam-Chennai · PETKODI TECH FUELS PRIVATE LIMITED · NEXTGEN GADGETRY EXTRUSION

Problem Analysis 1: Farmer Agent for Tailored Assistance:

What is the Problem?

Small-scale farmers across the world lack personalized, accessible, and timely agricultural support, especially in remote or resource-limited areas. Traditional extension services are limited in scale, language, responsiveness, and accessibility. This creates a digital divide, making it harder for farmers to:

Identify plant diseases or care needs.

Make informed decisions based on local weather, soil, or market data.

Access reliable guidance in their local language or preferred medium (voice/image).

Practice climate-smart agriculture for sustainable productivity.

Similarly, non-farmers (urban users) often struggle with plant identification and care, especially in hobby or home gardening scenarios.

How Does it Occur?

This problem arises due to multiple overlapping factors:

Digital Illiteracy: Many farmers are not comfortable with complex apps or text-heavy interfaces.

Language Barriers: Agricultural information is often not available in local dialects or languages.

Poor Infrastructure: Limited internet, electricity, or smartphone penetration in rural areas.

Fragmented Data: Weather, soil, and market data are siloed or inconsistent across regions.

Lack of Real-Time Help: Existing support is generic and not tailored to

Soil condition databases

Market price feeds

What is the Solution?

An AI-powered Farmer Agent that delivers multilingual, multimodal, and location-specific support through:

Core Features:

Voice, Text, and Image Input: For easy communication, especially in low-literacy settings.

Multilingual NLP: Understands and responds in regional languages and dialects.

Plant Identification via Computer Vision: Detects plant species and diseases using image input.

Personalized Crop Advisory: Integrates soil, weather, and market data to give tailored advice.

Climate-Smart Recommendations: Encourages sustainable practices and water-saving techniques.

Offline Mode: Basic features function without internet (e.g., image diagnosis, preloaded guidance).

How to Prevent the Problem in Future?

To prevent these problems from persisting or worsening:

Early Digital Training: Introduce basic digital skills training for farmers, especially women and youth.

Local Data Infrastructure: Collaborate with governments and startups to build regional soil/weather databases.

Community-Based AI Deployment: Deploy through local kiosks, SHGs, or farmer cooperatives.

Frequent Updates: Keep the AI trained on new crops, diseases, languages, and climate patterns.

Encourage Open-Source Collaboration: Share tools and datasets across NGOs, startups, and researchers.

Policy Support: Integrate AI tools into national agricultural support programs.

What This Problem Changes in Society (Future Impact)

If solved effectively, this problem's solution can bring transformative change:

For Farmers:

Increased Yields: Better decisions lead to more productivity.

Reduced Losses: Early disease detection and timely advice minimize crop failure.

Economic Growth: Access to market trends boosts profitability.

Social Equity: Women and marginalized farmers get equal access to knowledge.

Climate Resilience: Promotes sustainable practices, protecting long-term food security.

For Urban Users:

Improved Green Spaces: More successful home gardening and plant care.

Biodiversity Awareness: Educates people about native and exotic plants.

For Society:

Smarter Agriculture: Shift from reactive to proactive farming.

D**ata-Driven Policies:** Governments can plan better with real-time, ground-level insights.

AI for Good: Builds a real-world case of AI solving grassroots

Data-Driven Policies: Governments can plan better with real-time, ground-level insights.

AI for Good: Builds a real-world case of AI solving grassroots problems.

Requirements for the Solution
Technical Requirements
Multilingual NLP Engine

Supports voice and text in regional languages/dialects

Accurate speech-to-text and text-to-speech models

Multimodal Interface

Voice input/output

Text input/output

Image upload (for plant identification)

Computer Vision System

Image recognition models to identify plants, pests, and deficiencies

Disease detection based on leaf patterns and color

Real-Time Data Integration

APIs to access:

ocal weather data

Soil condition databases

Market price feeds

Crop calendars

Machine Learning Models

Personalized advisory systems based on crop, region, and season

Recommendation systems for care, treatment, and fertilization

Climate-Smart Advisory Module

Suggests sustainable, low-impact farming methods

Monitors environmental factors for crop selection and timing

User-Centric App Design

Simple UI for low-literacy useras

Offline mode support

Accessibility for women and marginalized groups

Data Storage

Secure storage of user history and preferences

User profiling for more accurate future recommendations

Societal Impact of the Solution

For Farmers

Empowerment: Gives knowledge at their fingertips in their language

Self-Reliance: Reduces dependence on unreliable middlemen or outdated advice

Higher Income: Better decisions lead to improved yield and sales

Resilience: Equipped to adapt to changing climate and market trends

For Women Farmers

Gender Equity: Accessible voice-based systems reduce gender gaps in advisory access

Inclusivity: Encourages participation in decision-making on farms

For Environment

Sustainable Practices: Reduces overuse of chemicals and water

Climate Adaptation: Promotes climate-resilient crop cycles and resource usage

For Society

Digital Literacy Uplift: Familiarizes rural populations with AI and digital tools

Agritech Growth: Spurs innovation in agri-AI startups and services

Data-Driven Policies: Real-time farming trends help governments create better support schemes

Food Security: Boosts productivity and reduces losses, contributing to national food safety

Problem Analysis 2: Digital Catalog Creation & Maintenance Agent

What is the Problem?

Millions of farmers, artisans, and small retail (kirana) store owners struggle to create proper product listings for digital platforms due to a lack of technical skills, descriptive writing experience, and consistency in catalog maintenance.

This limits their ability to reach online markets, compete on e-commerce platforms, or even maintain an organized local inventory digitally.

How Does it Occur?

This problem occurs due to several interconnected barriers:

Digital literacy gaps: Many sellers are unfamiliar with catalog tools or online product listing standards.

Language barriers: Difficulty expressing product features in market-ready language, especially translating from local languages to English.

Voice-only comfort: Many users prefer voice input due to literacy issues.

Lack of awareness: Sellers may not know how to describe products attractively or correctly (e.g., size, color, material).

Manual errors: Handwritten or unstructured data entry leads to mistakes and inconsistency.

Maintenance burden: Updating stock or prices is tedious and often neglected.

What is the Solution?

A voice- and text-enabled AI Catalog Agent that helps small sellers create and maintain high-quality product listings.

Core Features:

Voice and text input support in local languages and dialects

Automatic generation of market-ready product descriptions using AI

Product category mapping to match online marketplace standards

Guided input prompts to help complete missing product fields

Auto-translation from local inputs to English or other formats

Inventory update reminders and alerts for outdated or low-stock listings

How to Prevent the Problem in the Future?
Conduct digital catalog training through local NGOs, SHGs, and cooperatives

Integrate the tool into government MSME and Digital India schemes

Develop community-owned cataloging templates and standards

Ensure the tool is mobile-first and works on low-end devices

Set up feedback loops to refine AI suggestions based on user feedback

Deploy AI kiosks at panchayats or rural service centers for broader access

What This Problem Changes in Society (Future Impact)
For Farmers and Artisans:

Empowers them to sell directly to consumers via digital marketplaces

Increases income through better product visibility and descriptions

Builds brand identity and recognition even in rural settings

For Small Businesses:

Enables inventory digitization and easier stock management

Improves discoverability of products through digital listings

For Society at Large:

Promotes inclusive growth of the rural digital economy

Bridges the urban-rural divide in e-commerce and technology adoption

Demonstrates the democratizing power of AI for low-tech users

Requirements for the Solution: Digital Catalog Agent
Technical Requirements:

Speech-to-text engine: Must support regional languages and recognize rural accents

Natural language processing (NLP): Understands user intent and refines input descriptions

Generative AI model: Converts minimal input into full, marketable descriptions

Text-to-text translation: Handles accurate and meaningful translation between local languages and English

Voice assistant interface: Designed for users with low digital literacy; mobile- and kiosk-compatible

Catalog management dashboard: Simple interface for viewing, editing, and updating product entries

Cloud sync and offline mode: Works offline and syncs automatically when reconnected

Reminders and notifications: Alerts users about low stock, missing details, or seasonal catalog updates

impact of the Solution on Society
For Rural and Small Sellers:

Economic empowerment through online selling without middlemen

Greater market access, locally and globally

Enhanced professional image and business credibility

For Society:

Builds an inclusive digital economy

Preserves and promotes local and indigenous crafts

Improves supply chain transparency

Enhances gender equity by increasing women's visibility in digital commerce

Macro-Level Impact
Encourages local entrepreneurship and micro-enterprise development

Strengthens "Vocal for Local" and "Digital India" initiatives

Reduces rural unemployment and promotes self-sufficiency

Problem Analysis 3: Auto-Dock It!

What is the Problem?

Developers often face complex and inconsistent onboarding experiences when trying to run someone else's open-source project from GitHub.

Most repositories lack complete or up-to-date documentation, making it difficult to:

Identify required dependencies

Understand the tech stack

Set up the correct environment

Run the application without errors

This process can be time-consuming, especially for newcomers or those evaluating multiple projects quickly.

How Does it Occur?

This problem arises due to several factors:

Incomplete or outdated READMEs

Missing or unclear setup instructions

Hidden dependencies or services (e.g., database, message queue)

Language/framework diversity—every repo is different

Manual trial-and-error needed to get the project running

What is the Solution?
Build an LLM-powered tool that automates the process of understanding and containerizing GitHub repositories.

This tool should:

Clone a public GitHub repository

Use an LLM to analyze code structure, tech stack, and project type

Automatically generate a Dockerfile

Create a unified JSON or YAML config file with ports, environment variables, and commands

Optionally validate the container via a health check or test run

This enables plug-and-play container setups, removing the guesswork from running code.

How Will the Solution Make an Impact in the Future?
Zero-friction onboarding: Developers can spin up projects instantly, without needing manual instructions

Times New Roman

Accelerates innovation: Reduces time wasted on setup, encouraging faster prototyping and collaboration

Standardizes deployment: Enforces best practices for containerization across all projects

Boosts open-source adoption: Makes more repositories immediately usable, even for non-experts

Supports AI development: Allows LLMs and agents to run and test code automatically without human help

What Does This Change in Upcoming Days?
"Run Any Repo" becomes the norm: Developers will expect every repo to be container-ready

Shift in open-source standards: Repositories may start including auto-generated Docker configs by default

Codebase self-documentation: LLMs will increasingly explain code and setups, reducing the need for handwritten docs

One-click deployments: Integrations with cloud services will make launching code seamless

Requirements for the Solution
Core Functionalities
GitHub integration to clone and fetch repositories

LLM for code analysis (e.g., OpenAI GPT or Gemini)

Stack detection (language, frameworks, libraries)

Dockerfile generation logic

Config file generation (YAML/JSON for environment variables, ports, commands)

Optional support for Docker Compose for multi-service applications

Technical Requirements

OpenAI or Gemini API Key for model access

CLI or web interface for user interaction

Parsers for various dependency files like package.json, requirements.txt, pom.xml, etc.

Health check or simple test runner to validate Dockerized setup

Hosting service (optional) for a demo version such as Vercel, Netlify, or Render

Frontend (If Web UI is built)
Input field for GitHub repository URL

Preview of generated Dockerfile and configuration file

Download and run instructions

Logs for build/test results
Optional: Auto-generated explanation of the repo using the LLM

How to Prevent the Problem in the Future (and Present)?
Adopt AI-based onboarding: Encourage integration of tools like Auto-Dock It into CI/CD pipelines

Enforce containerization standards: Promote Dockerfiles or standardized templates for open-source projects

Educate developers: Raise awareness about the importance of standardized deployment methods

Contribute back: Enable users to submit generated Dockerfiles as pull requests to the original repo

Automate through GitHub Actions: Trigger container generation with every commit

Open-source the tool: Encourage the developer community to improve and extend the tool's capabilities

Automate through GitHub Actions: Trigger container generation with every commit

Open-source the tool: Encourage the developer community to improve and extend the tool's capabilities

Summary Impact
For Developers

Faster setup and less time wasted on configuration

Easier exploration and usage of open-source projects

For Organizations

Streamlined onboarding and development pipelines

Reduced DevOps burden for evaluating or integrating third-party code

For Hackathons and Collaborations

Rapid prototyping and testing of ideas

Simplified team collaboration across tech stacks

For AI Integration

Enables AI agents to autonomously understand and run real-world projects