LTIMindtree

Gardener Home Plant Logout

Customer Home Plant Logout

Garden Mentor

Empowering Plant Enthusiasts

A seamless platform built with React and .NET, hosted on Azure, connecting gardeners and plant lovers for personalized growth .

Introduction

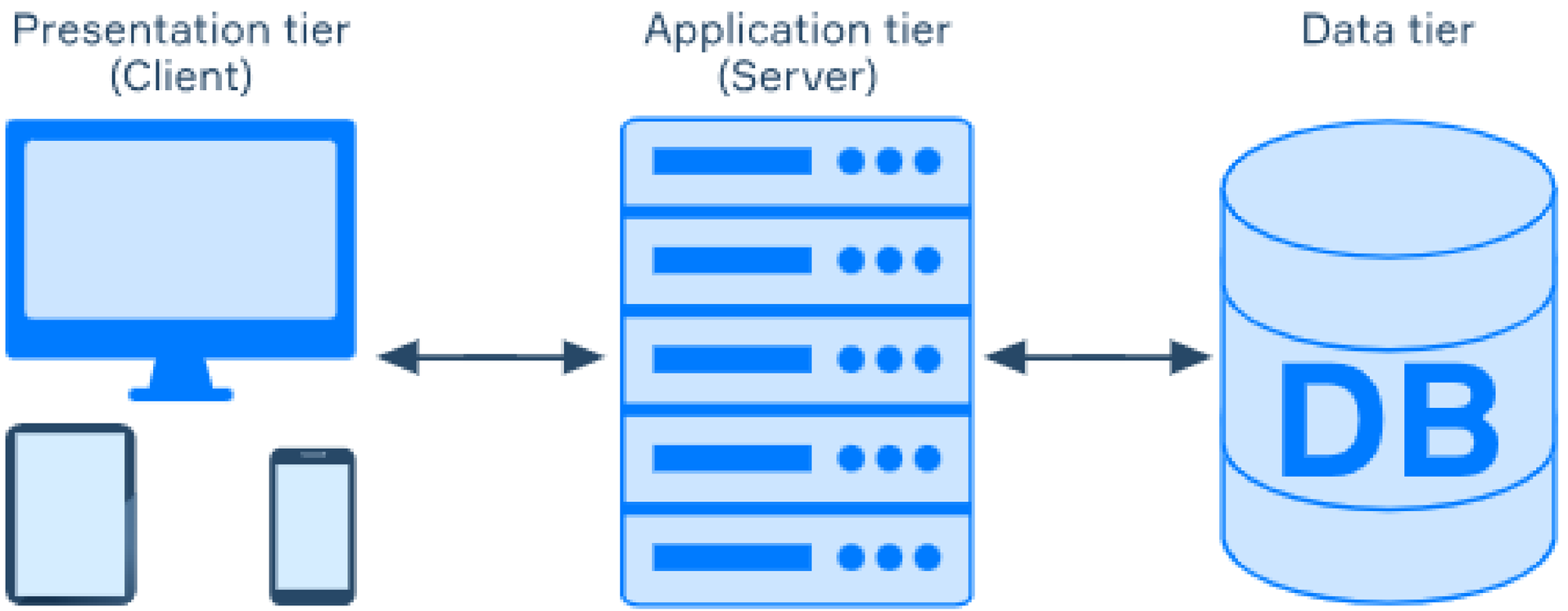
Many people are turning to gardening as a way to relax and connect with nature. However, not all has the knowledge to grow plants successfully. ***"Garden Mentor"*** creates a seamless platform that connects gardening enthusiasts with expert gardeners for curated plant list.

Technologies Used:

Frontend: *React.js*
Backend: *.NET Web API*
Deployment: *Azure Cloud*

Users can login/signup and based on their role will be directed to different pages- either gardener or customer where a gardener can view/add/delete/edit plants whereas customers can view/search plants.

3-Tier Architecture



Frontend React Overview

01

Framework/Library:

React JS Library is used to create the user interface of Garden Mentor.

02

Main Components:

Functional Components, Hooks for State Management and Routing for Navigation.

03

Key Libraries

Axios for API Calls and React Router for Navigation.

LTIMindtree

Login & Register Flow

KEY Features of User Authentication System

Login:

1. User Input: User enters email and password

2. API Call: Frontend sends POST request to /api/login

3. Authentication: Server checks credentials against the database

4. Token Generation: On success, JWT token is generated and returned

5. Response: Token is stored in frontend (localStorage/sessionStorage)

Register:

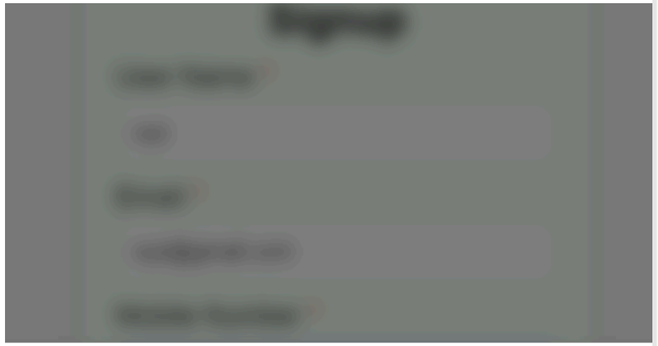
1. User Input: User enters username, email, mobile number, password and role.

2. API Call: Frontend sends POST request to /api/register

3. Validation: Server validates user input, checks if user already exists

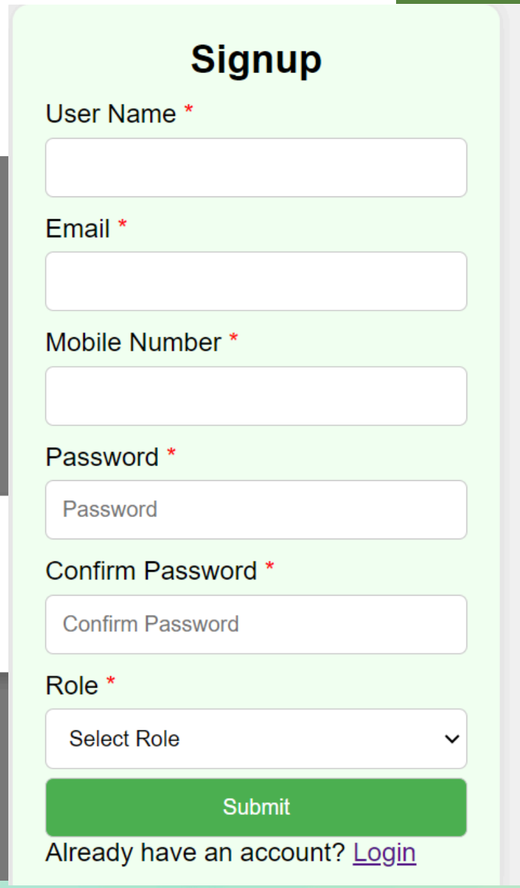
4. Data Storage: If valid, user is saved in the database

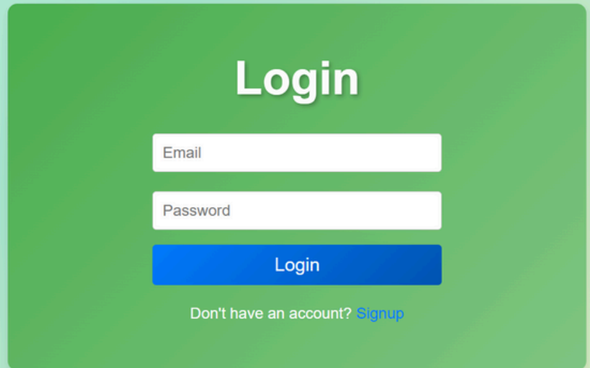
5. Response: Server returns success message



User Registration is Successful!

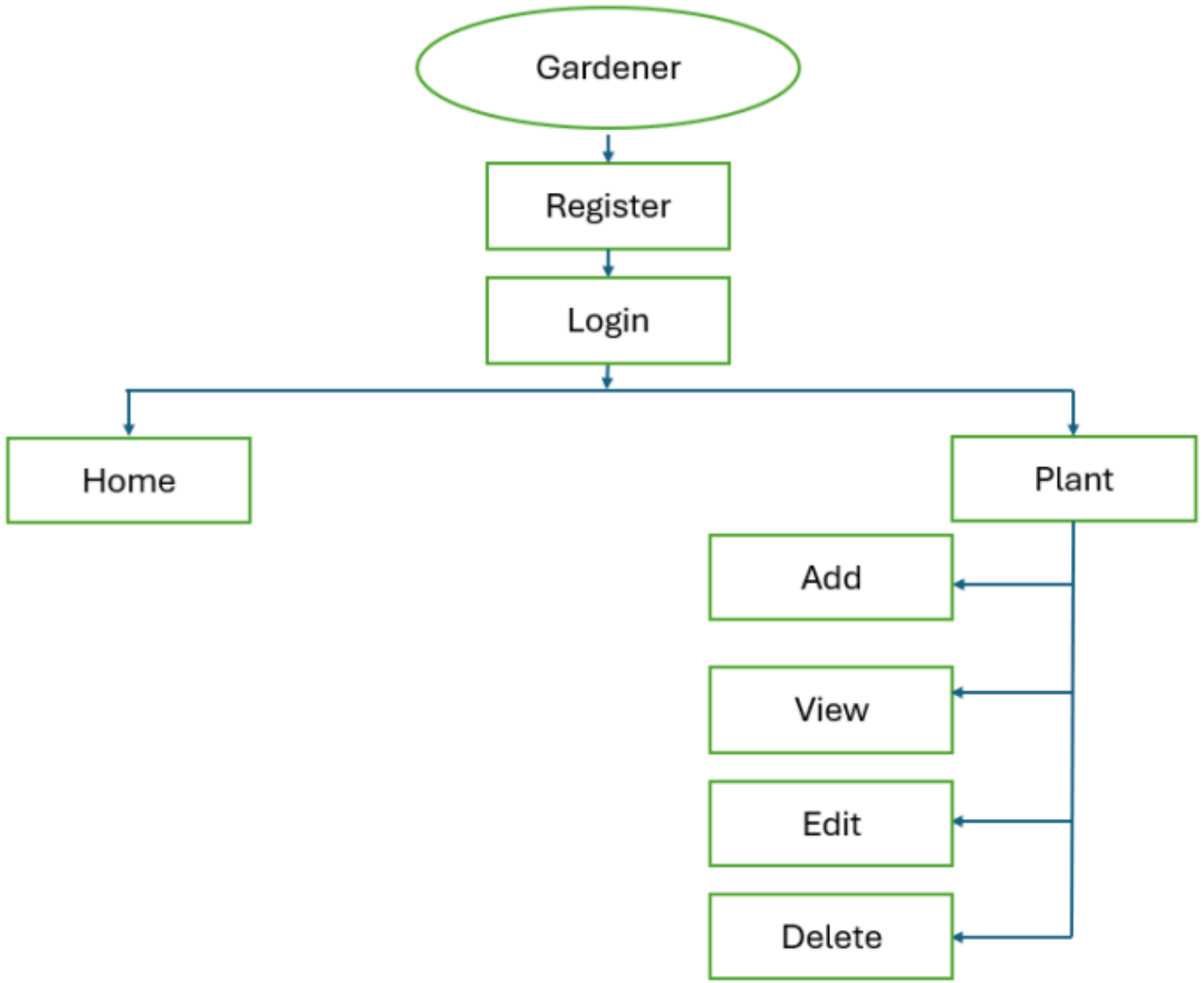
OK





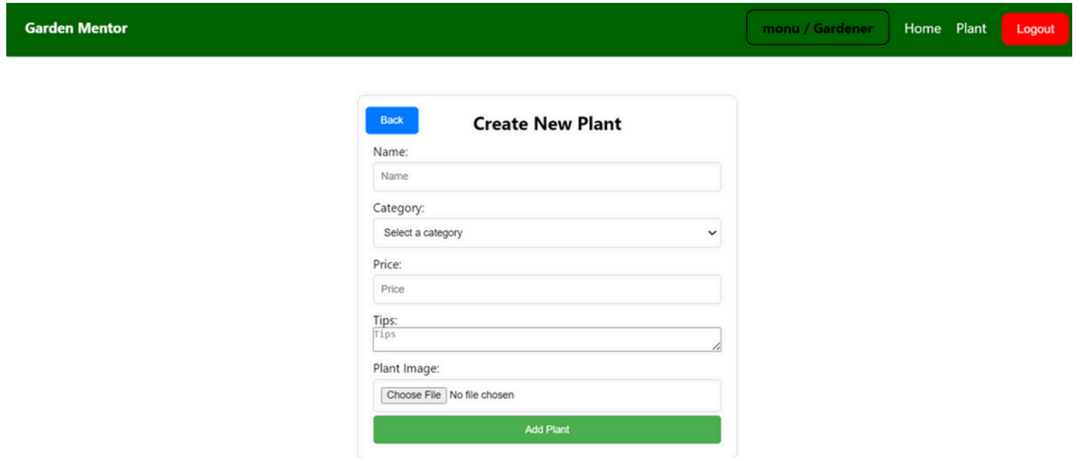
Gardener Flow (End-to-End)

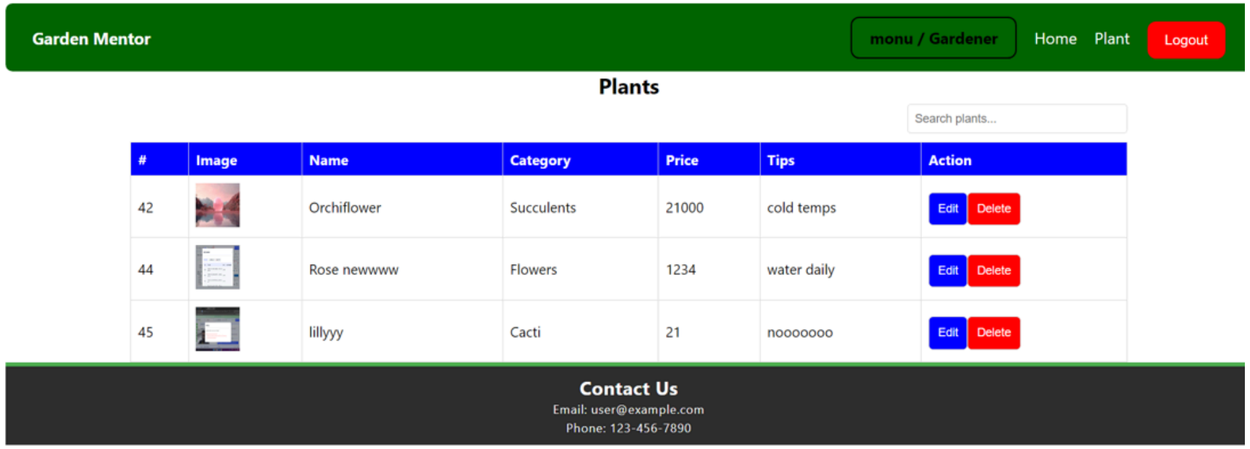
Gardener Flow Diagram:

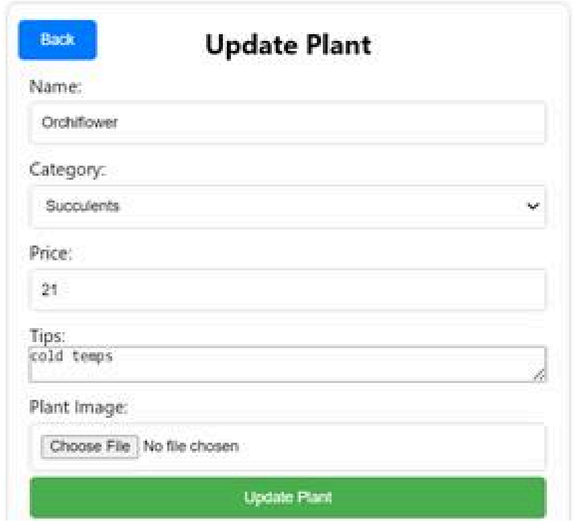


LTIMindtree

Gardener Component

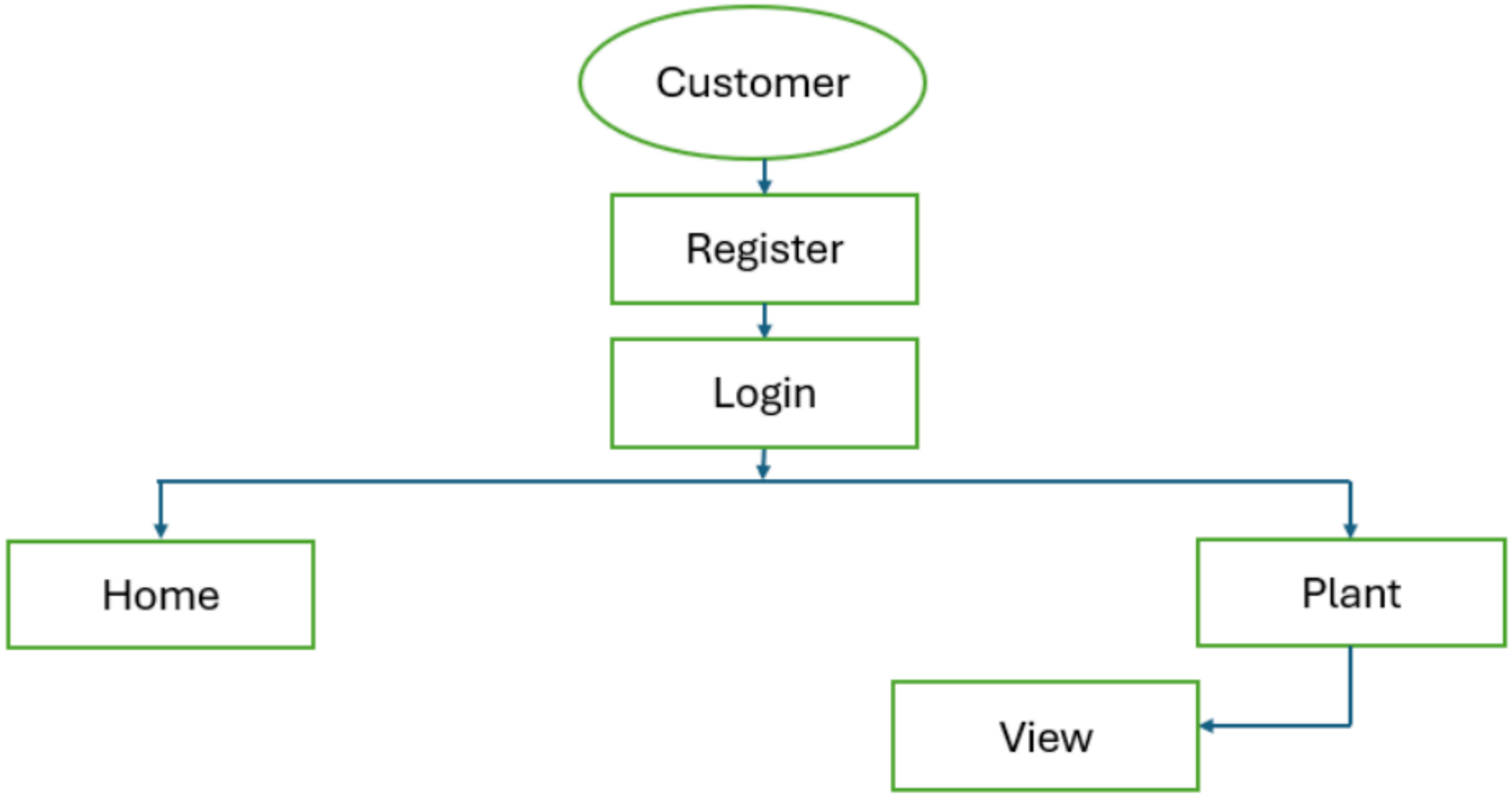






Customer Flow (End-to-End)

Customer Flow Diagram:



LTIMindtree

Customer Component

Backend .NET Overview

01

API Structure:

- *Used controllers to manage different endpoints.*
- *Each controller has services to process login and connect to the Db.*

02

Authentication and Data Processing;

Data is validated on the server-side, and responses are sent back in JSON format. using JWT tokens

03

Key API Endpoints

- **Plant**

GET /api/plants
POST /api/plants
GET, PUT, DELETE /api/plants/{plantId}

- **Authentication**

POST: /api/login
POST: /api/register

Azure Services Used

Azure App Service

1. Managed Web Hosting Platform: It allows developers to build, deploy, and scale web apps, APIs, and mobile backends using various frameworks (e.g., .NET, Node.js, Python).

2. Built-in CI/CD and Scaling: Supports continuous integration with tools like GitHub and Azure DevOps and offers automatic scaling to handle traffic demands.

Azure SQL Database

1. Managed Database as a Service: A fully managed relational database with automatic updates, backups, and high availability features.

2. Intelligent Performance Tuning: Built-in AI and machine learning capabilities help optimize query performance and manage workloads efficiently.

Azure Storage Account

1. Unified Storage Solutions: Provides access to blob, file, queue, and table storage, enabling data storage for various purposes like backups, media files, and log data.

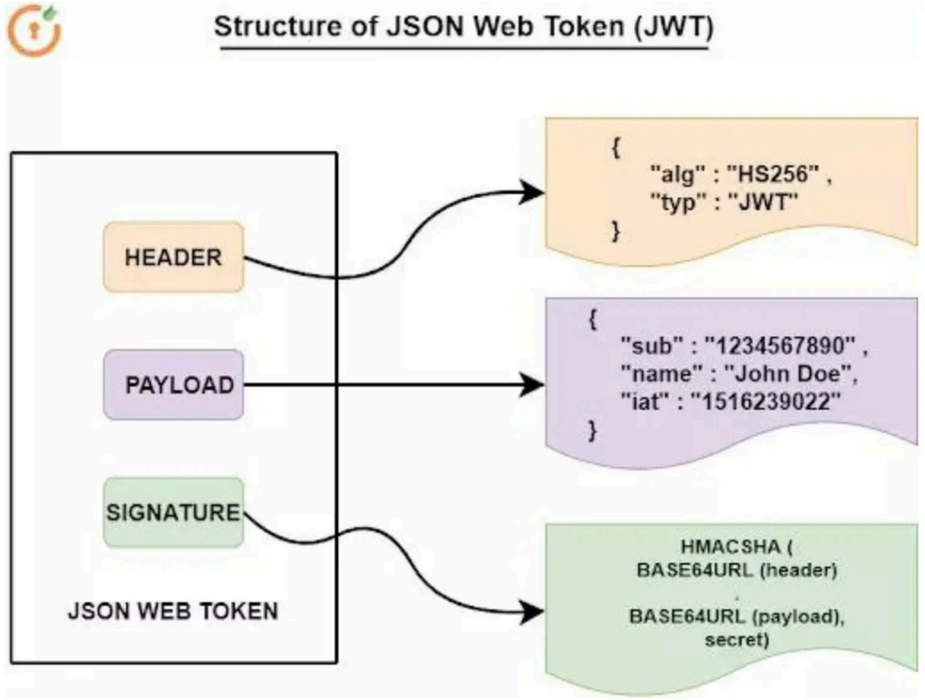
2. High Availability and Security: Offers redundancy options (LRS, GRS) and encryption to ensure data durability and protection.

Azure Key Vault

1. Centralized Secrets Management: Stores and manages sensitive information like API keys, passwords, and certificates securely.

2. Access Control and Auditing: Integrates with Azure Active Directory to enforce strict access policies and supports activity logging for security audits.

Security



Authentication & Authorization

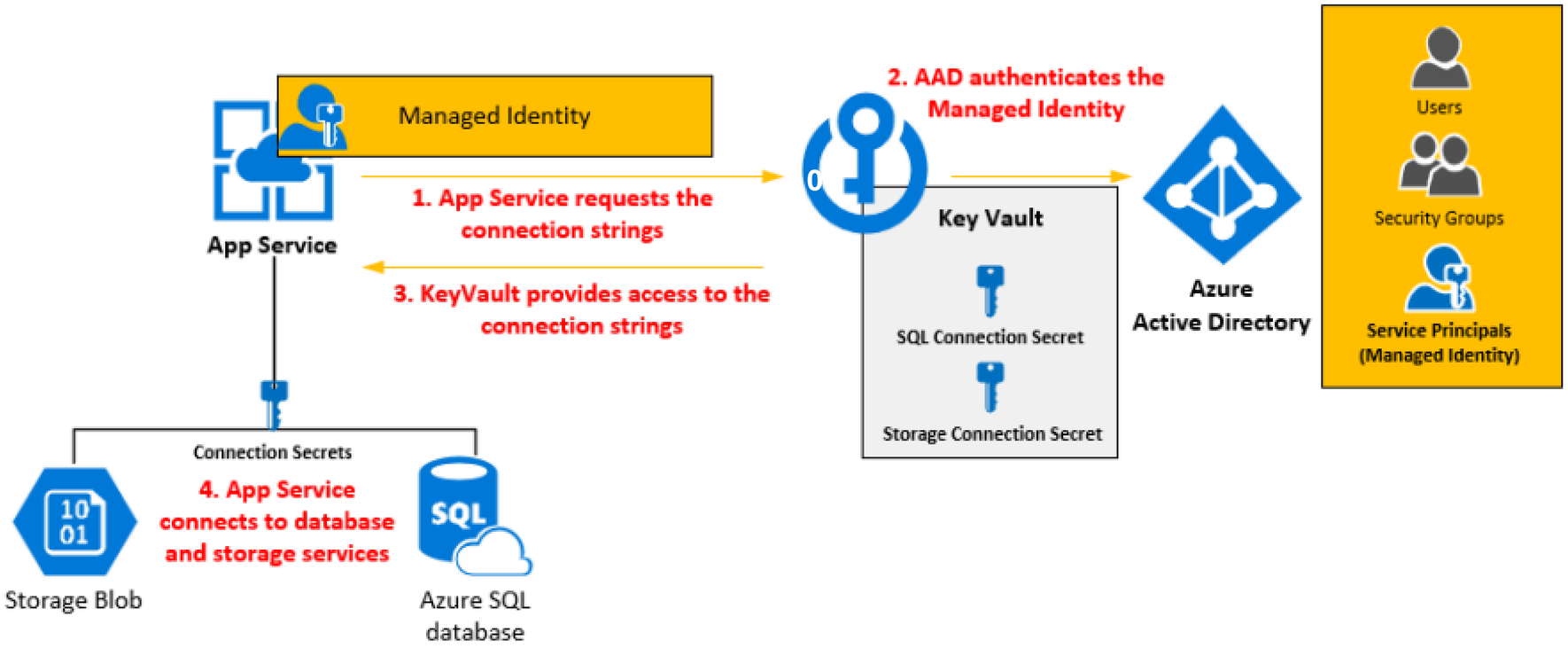
Used JWT (JSON Web Tokens) for user authentication and role- based access control

Azure Security Features

Used Azure Key Vaults to manage sensitive information like API keys, certificates etc.

Azure

Azure Deployment



Challenges

- *Initial struggles with API integration*
- *Ensuring smooth Azure deployment*

Conclusion & Future Scope:

Recap:

- Successfully built a full-stack application that allows users to connect with gardeners.
- Deployed securely and scalably using Azure.

Future Enhancements:

- Improve UI/UX.
- Add more personalization features.

LTIMindtree