IBMS Project Summary

Project Overview  
  
This project appears to be a full-stack web application named IBMS (possibly "Inventory and Billing Management System" or similar), consisting of a Django backend and a Next.js frontend. The project is organized into two main directories:  
  
- ibms\_backend/ — The backend, built with Django.  
- next-frontend/ — The frontend, built with Next.js (React).  
  
Backend: Django (ibms\_backend/)  
  
Structure  
  
- ibms\_backend/ibms\_backend/: Contains core Django project files:  
 - \_\_init\_\_.py: Marks the directory as a Python package.  
 - asgi.py, wsgi.py: Entry points for ASGI/WSGI servers.  
 - settings.py: Django project settings.  
 - urls.py: Root URL configuration.  
  
- ibms\_backend/inventory/: A Django app, likely handling inventory, billing, and customer management.  
 - \_\_init\_\_.py, admin.py, apps.py: Standard Django app files.  
 - models.py: Contains database models for inventory, billing, and customers.  
 - views.py: Contains view logic (API endpoints or web views).  
 - urls.py: App-specific URL routes.  
 - migrations/: Database migration files, indicating models for customers, bills, and bill items.  
 - tests.py: For unit tests.  
  
- manage.py: Django's command-line utility for administrative tasks.  
- venv/: Python virtual environment for dependency management.  
  
Frontend: Next.js (next-frontend/)  
  
Structure  
  
- app/: Main application directory (Next.js App Router structure).  
 - billing/, customers/, inventory/: Each contains a page.js file, likely representing pages for billing, customer management, and inventory.  
 - reports/, settings/: Directories for additional features (pages/components).  
 - layout.js: Defines the root layout for the app.  
 - page.js: The main landing page.  
 - globals.css: Global styles.  
 - favicon.ico: Site icon.  
  
- public/: Static assets (SVGs, images).  
  
- Configuration and Metadata:  
 - jsconfig.json: JavaScript/TypeScript configuration.  
 - next.config.mjs: Next.js configuration.  
 - package.json, package-lock.json: Node.js dependencies and scripts.  
 - postcss.config.mjs: PostCSS configuration.  
 - README.md: Project documentation.  
  
Key Features (Inferred)  
  
- Inventory Management: Backend models and frontend pages for managing inventory items.  
- Billing System: Models and pages for bills and bill items.  
- Customer Management: Models and pages for customer data.  
- Reports & Settings: Placeholder directories for reporting and application settings.  
- Modern Stack: Uses Django (Python) for backend and Next.js (React) for frontend, following best practices for separation of concerns.  
  
Development & Usage  
  
- Backend: Run and manage with Django's manage.py. Models and migrations suggest a relational database (likely SQLite, PostgreSQL, or MySQL).  
- Frontend: Developed with Next.js, using the App Router. Likely communicates with the backend via REST APIs or similar.  
- Virtual Environment: Python dependencies are isolated in venv/.  
- Node.js: Frontend dependencies managed via npm.  
  
Summary Diagram  
  
User --> Next.js Frontend --> Django Backend --> Database  
 |--> Static Assets/Public  
  
If you need a more detailed technical summary (e.g., specific models, API endpoints, or UI features), let me know!