Create App

`bench new-app library_management`

- **library_management:** App's source code are all here

  (*most of these dirs & files are empty in library_management)*

  (*in erpnext, these dirs contains tons of pre-made code files*)

  - **public**: Store static files that will be served from Nginx in production (*store css & js*) (*in erpnext, it stores a lot of pre-made scss, js, images, etc.*)
  - **templates**: Jinja templates used to render web views (*__init__.py)* (*in erpnext, contains tons of pre-made html, js, and .py for emails, pages, etc.*)
  - **www**: Web pages that are served based on their directory path (*empty*) (*in erpnext, contains pre-made html, js, and .py*)
  - **library_management:** Default Module bootstrapped with app (*__init__.py*)
  - **modules.txt:** List of modules defined in the app (*empty*) (*in erpnext, contains all standard modules such as Selling, Buying, Accounts*)
  - **patches.txt**: Patch entries for database migrations (*empty*) (*in erpnext, contains erpnext.patches.v*.* file*)
  - **hooks.py**: Hooks used to extend or intercept standard functionality provided by the framework (template for customization, like a Control Panel in Windows) (*in erpnext, the code for all of these are uncommented and pre-made*):
    - Basic App Information (import app version number & describe meta-data of the app)
    - Includes in <head> (include css & js files in header)
    - Home Pages (customizable default landing page)
    - Generators (auto webpage generation for each record of specific doctypes)
    - Jinja (add custom methods & filters to jinja env)
    - Installation & Uninstallation (can run custom code before/after installation/uninstallation)
    - Integration Setup & Cleanup (can setup/cleanup dependencies/integrations before/after install/uninstall)
    - Desk Notifications (config notifications in Desk UI)
    - Permissions (scripts to evaluate permission for doctypes)
    - DocType Class (override standard doctype classes with custom ones)
    - Document Events (run custom methods on specific events for documents)
    - Scheduled Tasks (set tasks to be run on schedules)
    - Testing (pre-checks if everything is set up correctly)
    - Overriding Methods (override whitelisted methods & doctype dashboards)
    - Request & Job Events (hooks before/after requests/jobs events)
    - User Data Protection (config data protection settings such as redacting specific fields)
    - Authentication and Authorization (hooks for authentication & authorization)
  - **pyproject.toml:** Specifies how your app is built, you can optionally add 3rd party Python dependencies here which will get installed when your app is installed. (*essentially a blueprint for the app, contains dependencies, configs, meta-data, etc. This file also might be referenced when running custom scripts/hooks. It can be crucial for post-installation, maintenance, updates, and recreating env*)