*Your answer - File Catalog*

**Software Deployment Plan**

*[Latest update: 2024-11-01 ]*

## 1. System Requirements

*Operating Systems:*

- **Windows:**

  Windows 10 and above (for x64)

- **macOS:**

  macOS 12, 13, 14 (for x64 and AArch64)

- **Linux:**

  Rocky Linux 8 and above (for x64 and AArch64), SuSE Leap 15 and above (for x64), Ubuntu Linux 24.0S (for x64), Oracle Linux 7 and above (for x64, and AArch64)

*Hardware Requirements:*

- Minimum 4GB of RAM
- 500MB of free disk space
- Intel i3 or equivalent processor

*Software Requirements:*

- Java JDK: Minimum JDK 23
- Maven: Version 3.6 or above
- Package Manager: An installation of Homebrew (Mac), apt (Ubuntu), or Zypper (SuSE) for dependencies

## 2. Deployment Strategy Summary

The **File Catalog** application is designed for local use, as a standalone Java Swing application packaged in a JAR file with Maven-managed dependencies.

*Deployment Artifacts:*

1. **Executable JAR file (**file-catalog.jar)

   contains the application, including all dependencies bundled via Maven.

2. **Installer Scripts:**

   Cross-platform installation scripts for setting up the environment.

   - *install.sh:* Shell script for Linux/macOS.
   - *install.bat*: Batch file for Windows.

3. **Configuration File:**

   Provides users with a configurable config.properties file for customization, stored locally.

*Distribution Kit:*

All components will be packaged in a ZIP file containing:

- file-catalog.jar
- Installation and configuration scripts
- README.md with instructions

**3. Installation Package Contents**

*3.1 Required Source or Compiled Files*

- *file-catalog.jar:*

  Main executable file with compiled source code and resources.

- *pom.xml:*

  Maven file managing dependencies and build settings.

*3.2 Required Third-party Components*

- Maven Dependencies: All dependencies specified in pom.xml will be included in the final JAR.
  - Common libraries may include Swing components, Apache Commons, etc.

*3.4 Required Graphical Assets, Configuration, and Other Non-program Files*

- *Graphical Assets*:
  Icons for the UI, stored in src/images/.

- *Configuration*:
  config.properties for user-defined settings, located in src/resources/config.

*3.5 Documentation Files to be Provided*

- README.md: Instructions for installation, basic usage, and system requirements.

*3.6 Development Files and Components to Exclude*

- *Source Code*:

  All source code files (e.g., src/) excluded from final distribution.

- *IDE Files*:

  Exclude IDE-specific files (e.g., .idea directory, any temporary build files).

- *Testing Artifacts*:

  Exclude files generated in target/ or out/..

**4. Additional Code Required for Deployment**

*Shell Script for Linux/macOS* (install.sh):

- Sets up the required environment locally, checks for Java and Maven installations, installs any missing dependencies, and configures paths.

Batch Script for Windows (install.bat):

- Provides the same installation process for Windows users, ensuring a consistent installation experience.

*Configuration Setup Script* (config-setup.sh):

- Allows users to customize settings and sets up config.properties with default values. Ensures proper permissions for config.properties where needed.

**5. Deployment Tasks**

*Prepare the project for packaging:*

- ☐ **Verify Code Quality:**
  Run tests to ensure no issues are present in the code.
- ☐ **Commit All Necessary Code**:
  Ensure the latest code is in the repository.
- ☐ **Organize Resources**: Ensure resources like images, configuration files, and scripts are correctly organized.
- ☐ **Update pom.xml for Build Settings**:
  Double-check for dependencies and ensure plugins (like the Maven Shade Plugin) are correctly configured.
- ☐ **Set Version Numbers**:
  Ensure the project version is updated in pom.xml if necessary.

*Set Up the Maven Build Process:*

- ☐ **Maven Shade Plugin Configuration**:

  Ensure the plugin is set up to package dependencies in an executable JAR.
- ☐ **Build Verification**:
  Run mvn clean package to confirm that the build process completes without errors.
- ☐ **Testing the JAR**:
  Run java -jar target/file-catalog.jar to verify it works as a standalone JAR.
- ☐ **Error Logging Verification**:
  Check that logging works as expected within the packaged JAR to capture errors.

*Create Distribution Kit:*

- [ ] **Assemble Required Files**:
  Collect file-catalog.jar, README.md, and other necessary documentation and scripts.
- [ ] **File Structure Check**:
  Ensure the directory structure in the distribution kit matches what's described in the README.
- [ ] **Configuration Documentation**:
  Ensure config.properties includes example values and detailed instructions for setup.

*Create Installer Scripts:*

- [ ] Write platform-specific installer scripts (**install.sh**, **install.bat**).
- [ ] Verify both scripts work correctly by testing them on the respective operating systems.

*Configuration Setup Script*

- [ ] **Permissions Check**:

  Ensure config-setup.sh configures read/write permissions appropriately.

- [ ] **Platform-Specific Adaptations**:

  Account for platform differences, especially in file paths and permissions on Linux/macOS.

- [ ] **Sample Configuration**:

  Include sample values in config.properties for users to customize.

*Package Distribution kit*

- [ ] **Compression Test**:
  Ensure the ZIP file includes the correct folder structure and all required files.

*Testing and deployment:*

- [ ] **Cross-Platform Testing**:

  Execute on Windows, MacOS, and Linux systems as part of acceptance testing.

- [ ] **Update Documentation**:

  Document any required adjustments in README.md and other guides based on testing feedback.

- [ ] Deploy distribution file

**6. Deployment Test Plan**

| Test Case | Test step | Expected Result |
|---|---|---|
| Verify Installation Script Execution | Run the installation script (install.sh on Linux/MacOS, install.bat on Windows) | Script sets up file in correct directory and install necessary dependencies |
| Script Error Handling | Run installation script without dependencies like Java or Maven installed | Script should detect missing dependencies, display error messages, and guide user to install required dependencies. |
| Check Java SDK 23 Installation | Run java -version | Output shows Java version 23 |
| Platform-Specific Settings Verification: | Check JAVA_HOME and advanced settings on Windows | JAVA_HOME and other environment variables should be configured correctly. |
| Check Maven Installation | Run mvn -version | Correct Maven version is installed and shown. |
| Validate Application Directory Structure | Look at application directory to see all files | Application directory contains file-catalog.jar, config/config.properties, and scripts folder with config scripts |
| Configuration File Setup | Run the configuration setup script (config-setup.sh or config-setup.bat) | Creates config.properties file, and settings can be customized per user needs. |
| Configuration File Validation | Enter invalid values in config.properties | Script detects incorrect values, prompts the user to correct them, and does not proceed with invalid configuration. |
| Basic Application Functionality | Verify UI interaction, file cataloging, and other functions | Application responds correctly to all interactions, updates the catalog, and handles search inputs as expected. |
| Test Dependency Isolation (JAR Packaging) | Verify that all dependencies are packaged within file-catalog.jar | All dependencies are included, ensuring the application does not require external dependencies. |
| Documentation and Help Command | Verify the clarity of instructions in README.md | Documentation is clear, concise, and covers all installation, configuration, and usage steps comprehensively. |
| | | |

| | | |
|---|---|---|
| Uninstallation Process | Remove the application files | No residual files or dependencies remain after uninstallation, ensuring a clean system state. |