The Book Assembler

Test Plan Results

Author: Kyrylo Krysko

CONTENTS

INTRODUCTION

Due to the luck of expertize and very limited scope of the project standard automated testing techniques was not implemented. Developer performed list of manual tests defined at requirements and design stages.

1. LAUNCH BOOK ASSEMBLER

Test Case Description: User double click launcher file in the application install directory.
Test Case Input: Application install directory.
Test Case Expected Results: Main window appears.

This test was performed many times starting the first development stages till the finalization of the project. Issues that was found and fixed:

- Launch for the first time (no database file exists) but settings file contain previous project record. The initialization function init_ui() was restructured so database created before assembler tries to get any data from database.
- Failed to run when deployed (because of missing libraries in deployed Python). Resolution: update deployed python with necessary packages (PySide and Roprtlab)
- Launcher has absolute path, deploying/moving codebase broke launcher. Provided relative paths to fix this.

2. CREATE A NEW PROJECT

Test Case Description: With the Book Assembler launched, user click “Create/Set project”. File Open Dialog appears, user select a root folder for a new project: “C:/Users/kko8/OneDrive/projects/master/CIS575/projects/workbook”
Test Case Input: The Book Assembler.
Test Case Expected Results: The Book Assembler shows current project name and path in UI.

Catch issue with recording current project o the settings file: if user close File Open Dialog the settings JSON file corrupted. Fixed in settings module.

3. ADD NEW PAGES TO A PROJECT

Test Case Description: User copy pages from test project (provided by client) to a new project. With the Book Assembler launched, user click “Reload Pages” button.
Test Case Input: New pages copied to <root project>/pages/jpg folder.
Test Case Expected Results: The Book Assembler shows list of pages in UI. The new pages should be added to database tables.

Works as expected but during tests I come up with some improvement ideas for the next iterations:

- Need to implement validation of the page naming pattern. Despite we state in requirements that we expect valid input page names, the user errors are likely will happen. Nice to skip pages that does not meet naming convention.
- Ability to delete pages from the database. If user accidently copied irrelevant files we should be able to clean up database.
- Ability to automatically generate full list of blank pages with proper names. Can save some time and avoid user naming error.

4. SET DIFFERENT PROJECT AS CURRENT

Test Case Description: User click “Create/Set project” button and select another project in File Dialog.
Test Case Input: Another project with data
Test Case Expected Results: The Book Assembler shows list of pages for a new project in UI

This test reveal critical design issue. We don’t register projects in the database, all pages located in one table and we are getting pages from table by the name. Hence if different projects will share the same page names the book assembler would not work correctly. Need to introduce “projects” table and link pages to projects.

Also, the pages proportions are static, so any image with width/height = 1.295 (proportions of images from the first test project) will be distorted. Need to determine image proportions dynamically for every page shown in UI.

Those fixes left for the future iterations.

Another option to consider: get rid of SQLite database and save all necessary data in JSON folder for every project individually. More robust and reliable solution that will be enough considering small amount of potential data.

5. PREVIEW PAGE

Test Case Description: User selects the page in the list of pages
Test Case Input: Project with pages.
Test Case Expected Results: The Book Assembler shows page version 01 image in UI

- If we do not have 01 version but we have next versions The Book Assembler shows no images. Updated to show any next version if 01 version does not exists.
- If we switch to another project with different page dimensions the image will be displayed with a wrong proportions.

6. ADD PAGE DESCRIPTION

Test Case Description: User selects the page in the list of pages, double click description field and enter text.
Test Case Input: Project with pages.
Test Case Expected Results: The Book Assembler shows page description text in UI, corresponding database record created.

Utilized PySide Model View functionality for viewing/editing data, no issues encountered.

7. PREVIEW PREVIOUS/NEXT PAGE VERSION

Test Case Description: User selects the page in the list of pages, and press +/- buttons
Test Case Input: Project with pages.
Test Case Expected Results: The Book Assembler shows the +1/-1 page version images in UI.

Test reveal that on edge cases (try to show previous page of a minimal version or try to show next page of a maximum version) Assembler stuck in infinite loop while trying to wrap around existing sequence and switch to the first/last existing page. Fixed by getting rid of wrapping functionality.

Future improvement consideration: do not let assembler to display empty page if it is not exist, stick with existing pages.

8. PUBLISH CURRENT PAGE VERSION

Test Case Description: User selects the page in the list of pages, press +/- buttons to set desired version, press “Publish Current Page Version”.
Test Case Input: Project with pages.
Test Case Expected Results: The Book Assembler shows the published version in UI for selected page. Database record should be created.

Working as expected

9. SENT PAGES TO LAYOUT

Test Case Description: User clicks ”Sent Published Versions” button.
Test Case Input: Project with pages.
Test Case Expected Results: The Book Assembler copy all pages of published versions to “to_layout” folder. Page version should be cut from the file name. If page is unpublished, use 01 version.

The page version sent to layout was taken from UI that leads to sending incorrect versions sometimes. The sent_pages() function was updated to sent published versions.

10. GENERATE PDF FILE

Test Case Description: User clicks ”Generate PDF” button.
Test Case Input: Project with pages.
Test Case Expected Results: The Book Assembler creates a PDF file in “pdf” folder. PDF should contain images of published versions. If page is unpublished, use 01 version.

Initial design of PDF creation module involved utilizing the Settings module to get extra data. This leads to errors in cases when we switch projects. Fixed by avoiding using Settings module in PDF module and providing necessary data to PDF module from Assembler as arguments.

PDF module inserts page number on top of existing images for preview purposes. If image dimension changes the page number no longer located at a right place. Need to be fixed together with “Preview Page” function.

11. LAUNCH DOCUMENTATION

Test Case Description: User clicks in main menu: Help > Documentation.
Test Case Input: The Book Assembler Main Menu.
Test Case Expected Results: The Book Assembler opens application documentation in default web browser.

Path to documentation was absolute, deploying/moving codebase fails to load documentation. Provided relative paths to fix this.