The Book Assembler

Design Document

Author: Kyrylo Krysko

CONTENTS

1. Introduction 3
1.1 Purpose 3
1.2 Scope 3
1.3 User Workflow 3
1.4 Application description 3
1.5 Definitions 4
2. System Architecture 4
2.1 Application stack 4
2.2 Application components 5
3. Data Model 5
3.1 Data flow diagram 5
3.2 Class diagrams 5
3.3 Database Schema 6
4. User Interface Design and Functionality 6
5. Component Design 7
5.1 Settings module 7
5.2 Database module 8
5.3 PDF module 9
5.4 Assembler 9
5.4.1 Book Model 9
5.4.2 Assembler 9
6. Test Plan 9
6.1. Overview 9
6.2 Test units 10

1. INTRODUCTION

1.1 PURPOSE

This document presents the detailed design of the Book Assembler application. It outlines the software architecture, user interface design, component design, and data models. The Software Design Document (SDD) serves as a comprehensive guide for developers to implement the system as per the specified requirements and design.

1.2 SCOPE

The Book Assembler application is a standalone desktop application designed to manage project data in the book developing process.

The scope of this design document covers the software's overall architecture, data model, user interface design, and the detailed design of individual components and modules.

This is the Software Requirements document summary

1.3 USER WORKFLOW

In order to understand the software requirements we need to understand the workflow of book creation activity. The process of book developing includes such **phases**:

- Concept,
- Design,
- Layout,
- Printing.

During **concept** phase the Book Author creates the draft version of the book that consist of a page sketches. Once concept is done each page is saved as a jpeg file and sent to Illustrator.

At the **design** phase illustrator creates the final illustrations and return them back to author for review and storage. Author place illustrations into the same folder with pages. Author decides when page is finalized and publishes the finalized version (record the approved version to the database).

Once all Illustrations are finalized, they being send to **layout** artist who creates a Print PDF file for **printing**. This file is sent to a printing facility to produce the circulation of the book.

The book development process is not linear, each page goes through multiple rounds of revisions, and hence each page file has multiple versions.

The pages that Author sending to Layout Artist does not contain version component in the file name, which allows automatic update of all pages to the final in Layout Software

1.4 APPLICATION DESCRIPTION

Client (book author) wants to get a system that will automate: managing of pages and versions, sending approved pages to layout artist, creation of PDF file containing all pages. System should be able to handle any amount of projects (meaning that one book is a one project).

The Book Assembler will ask user to create new or select existing project. Within the selected project Assembler scans the folder with pages and record each page to the database relying on page name (see details in section 6: Data Requirements). The list of all existed versions will be shown in UI as a table where for each page user can also see the published version, sent version and page description.

User can select the page from the list and see the page image in UI. The system will display the published version. If nothing was published, the first (01) version will be shown.

User can observe previous or next versions of selected page pressing “+” or “-“ buttons.

User can record (publish) the current displayed version as approved by pressing “Publish Current Version” button.

User can sent all or selected pages to a layout directory with “Sent Published versions” button. System will copy image files without the version component in the name to a predefined folder. Those images would be used at layout phase.

1.5 DEFINITIONS

Term	Definition
The Book	Paperback printed book that consist of pages with illustrations
Book Author	Person who come up with the book idea, develop overall structure and detailed description of the content. Book Author draws sketches of all pages and sends them to illustrator.
Current Project	Application should support any number of projects it can work with but it can work only with one project at a time. Current project means project that user is working on during the session. Project equals book.
Illustrator	Person who takes the sketch of each page and produce the final illustration.
Layout Artist	Person who takes all final illustrations and assembles a digital version of the book (PDF) for printing in a printing house.
Page	The basic entity of the book, each page correspond to one page file on disk. Each file could have multiple versions.
Sketch	Page characteristic, rough version of a page drawing that shows what should be on the illustration.
Illustration	Page characteristic, refined sketch, the final version of the page that would be printed.
Publishing of a Page	Recording to the database which version of the page is a final illustration
Sending a Page	Each published version can be sent to layout artist. We should keep tracking of sent versions.
Preview PDF	The PDF file of the book for users observation produced by Book Assembler
Print PDF	The PDF file of the book for printing produced by Layout Artist
System Analyst	Software Development manager person, $ 75 per hour rate
Developer	Software Developer, $ 50 per hour rate

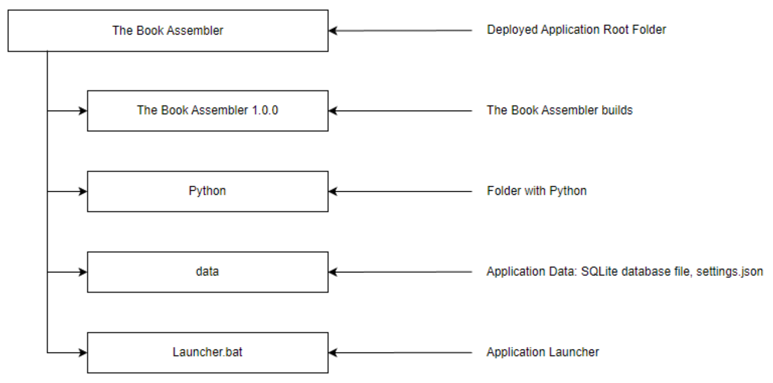
2. SYSTEM ARCHITECTURE

2.1 APPLICATION STACK

The Book Assembler will be implemented with Python 3.7 for Windows 10 OS. In addition to built-in SQLite module for database we will need to install PySide2 for UI. The application will be delivered as a package with Python included (that should eliminate all possible errors with missing dependencies).

The Book Assembler will be stored on GitHub repository. The Build and Deploy mechanism should be created to deploy the Book Assembler versions to a client.

Application deploy structure:


The Book Assembler
Deployed Application Root Folder
The Book Assembler 1.0.0
The Book Assembler builds
Python
Folder with Python
data
Application Data: SQLite database file, settings.json
Launcher.bat
Application Launcher

2.2 APPLICATION COMPONENTS

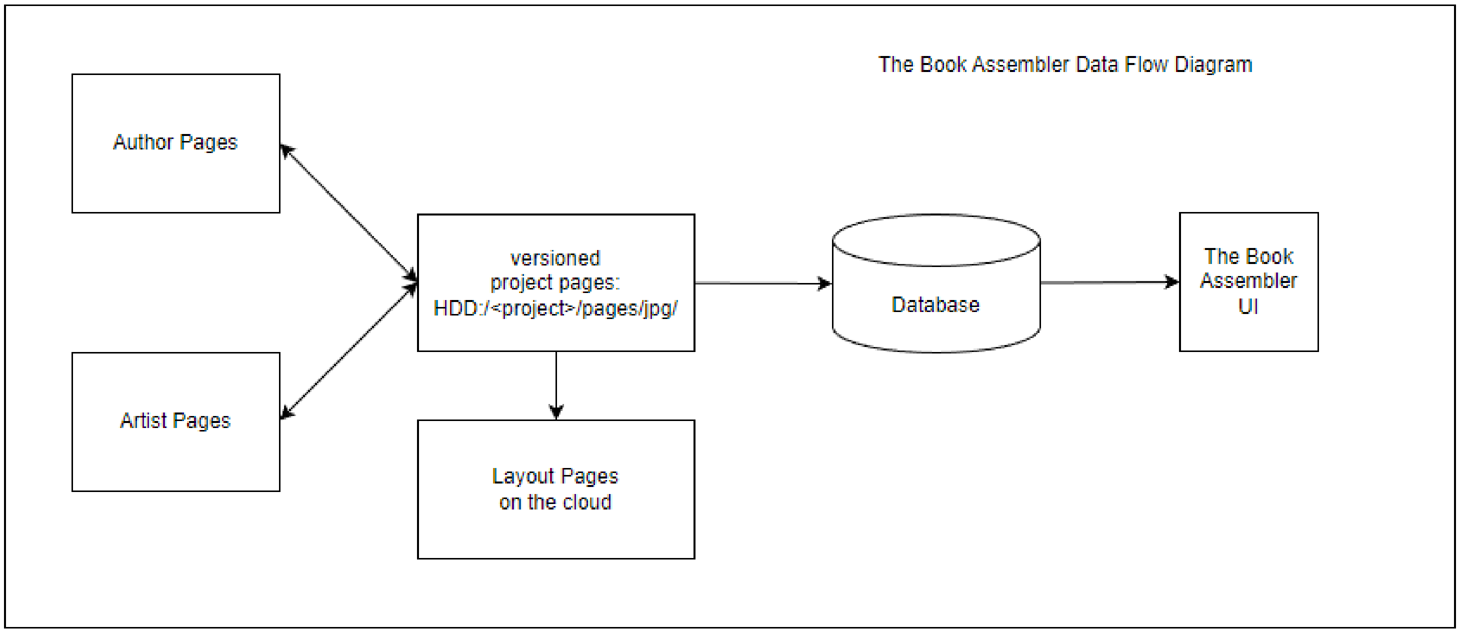
The Book Assembler components:

- Documentation (html files compiled from *.rst files with Sphynx theme)
- The Book Assembler database (data.db) and settings (settings.json) files located in “data” folder.
- User Interface Python file of main window, compiled *.ui file made in QTDesigner.
- The Book Assembler launcher, *.bat file that runs application python file with provided Python
- The Book Assembler application *.py file
- The Book Assembler python modules:
    - Database, for database operations
    - Pdf, for pdf document generation
    - Settings, for loading/modifying application settings

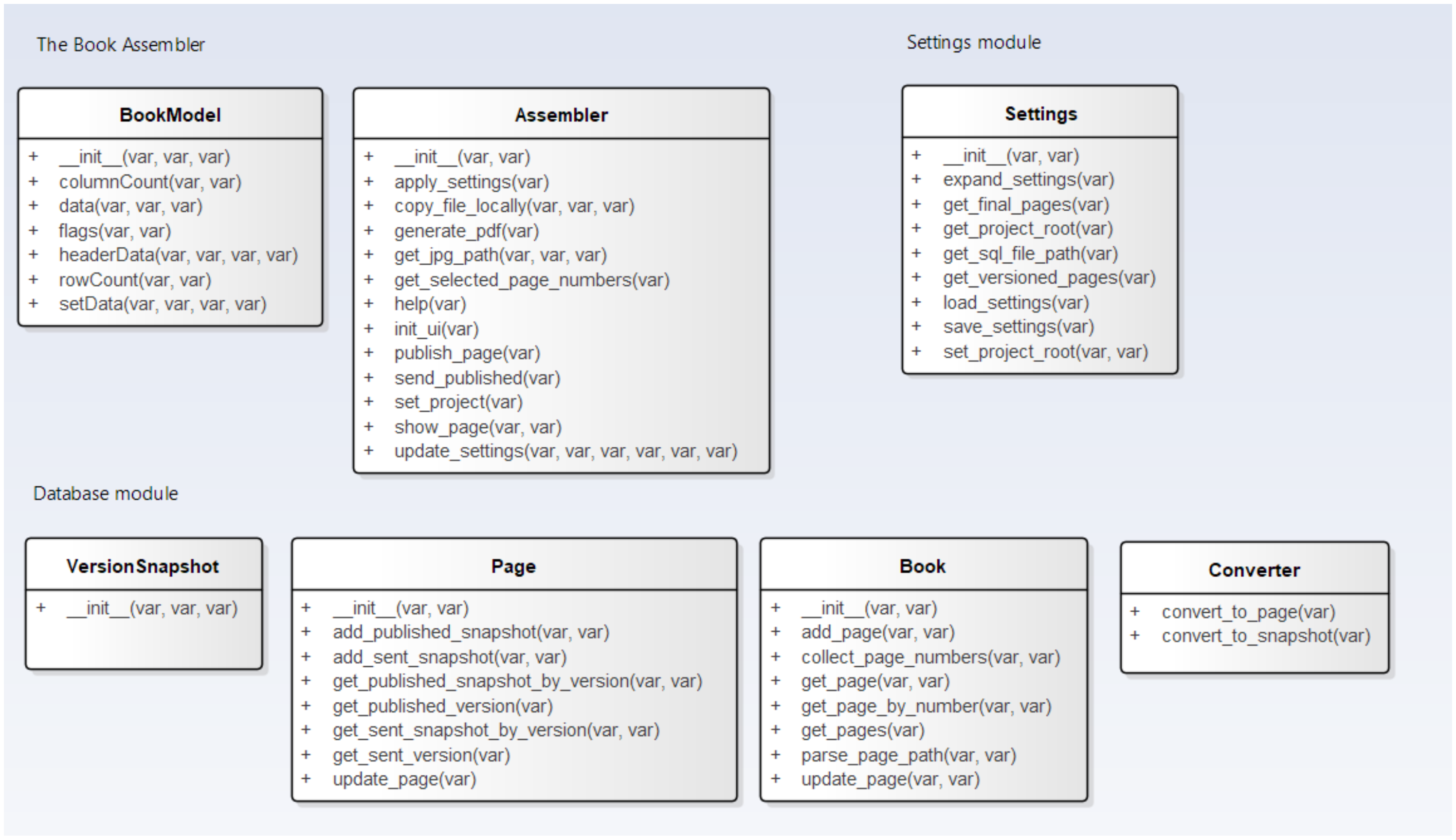
3. DATA MODEL

Define the data model in more detail, including data types, relationships, and constraints. For example, a task object could include fields like an ID, title, description, priority level, due date, and completion status.

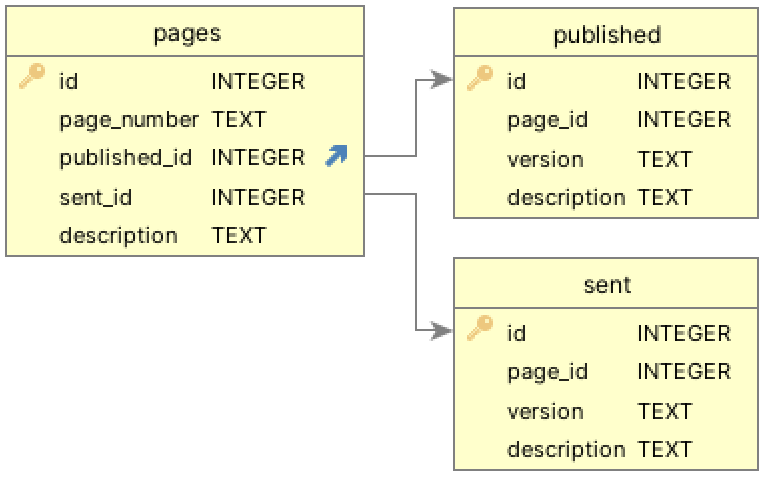
3.1 DATA FLOW DIAGRAM


The Book Assembler Data Flow Diagram
Author Pages
Artist Pages
versioned project pages: HDD:/<project>/pages/jpg/
Layout Pages on the cloud
Database
The Book Assembler UI

3.2 CLASS DIAGRAMS



3.3 DATABASE SCHEMA



4. USER INTERFACE DESIGN AND FUNCTIONALITY

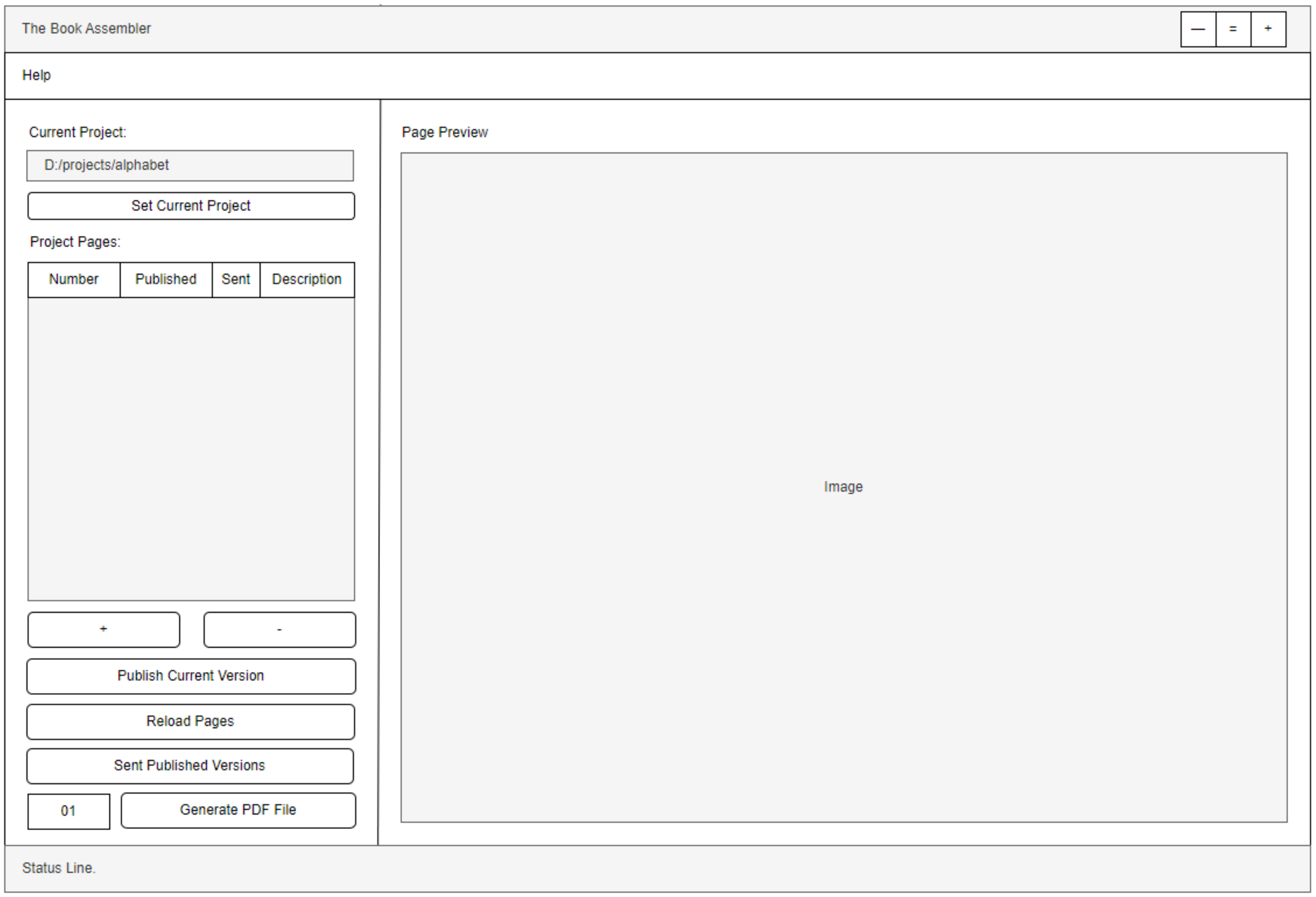
The Book Assembler has one main window and has standard Windows OS attributes: Title Bar, Menu Bar, Work Area and Status Bar.

- Title bar contain the application title “The Book Assembler” and minimize, maximize and close buttons.
- Menu bar contains Help > Documentation item
- Status bar used for displaying messages for the application user regarding performing critical actions.
- Work Area consist of two main areas:
  - Project Functions:
    - “Current Project”, QLabel and QLineEdit widgets. Displays path to the active project on HDD.
    - Set Current Project, QPushButton widget. Allows user to change current project to another one. Pressing this button will launch OpenFolder dialog where user should select project root folder.
    - “Project Pages”, QLabel and QTableView widget. Provides the list of existing pages and display page name, page published version, page version that was sent to layout artist and page description. Page description can

be edited by user by double-clicking the cell and entering text. Other columns are defined by project data itself and cannot be edited directly.

  - The “Minus” and “Plus” buttons, QPushButton widgets. Increases/decreases displayed page version.
  - “Publish Current Version” QPushButton widget. Records current version (selected/modified with the Plus and Minus buttons) of page to the database as final. Published version will be used for PDF createion and sent to layout artist.
  - “Reload Pages” QPushButton widget. Allows user to reload project data form HDD in case new pages were added to the folder during current session of the application.
  - “Sent Published Versions” QPushButton widget. Copy published versions of all existing pages to the folder in the cloud accessible by layout artist. Copied pages do not have version component in their names.
  - “PDF version” QlineEdit and “Generate PDF Flie” QPush button widgets. Generates PDF file with all published versions of all existing pages. PDF version define PDF file version: <PDF name>_PDF version>.pdf
- Image display. Here user can view selected page versions.

The UI build on Model View Controller design pattern, e.g. we use Model View widgets that allows to display actual data from database and operate python objects instead of strings.



5. COMPONENT DESIGN

The design of each major components in the Book Assembler

5.1 SETTINGS MODULE

Setting module stores/reads/modifies application attributes in JSON file that needs to be accessed from other modules:

- Current Project path,
- Relative paths to application data: SQLite database data.db file, settings.json file,
- Relative paths to project data: Pages, PDF files, Layout files.

Setting module should resolve relative paths to absolute paths depending on Current Project or Current Build number.

Settings module contains Settings class.

Attributes:

- assembler_root: string pathThe Book Assembler deployed root
- settings_file: string path to settings.json
- setting_data: hash map raw data loaded from settings.json
- current_settings: hash map resolved data for current project

Methods:

- expand_settings: reads data from setting.json, saves to settings_data attribute, resolve project paths, save to current_settings attribute
- Methods to get and return all required paths: get_sql_file_path, get_project_root, etc.

5.2 DATABASE MODULE

The database module responsible for CRUD operations and database creation (creating data.db and building all necessary tables when application is launched for the first time)

Database modules classes:

- Page: represents page objects
    - Attributes:
        - Id: database id
        - name: page name
        - published_id: id of published version record from “published” table
        - sent_id: id of sent version record from “sent” table
        - description: string user description of page
    - Methods:
        - update_page: edit page data in the database
        - get_published_version: returns published version number for current page
        - get_sent_version: returns sent version number for current page
        - add_published_snapshot: record published version of current page
        - add_sent_snapshot: record sent version of current page
- Snapshot: class representing published or sent version
    - Attributes:
        - Id: databse id
        - page_id: id of published page
        - version: published version
        - description: string field for notes

- Book: represents book object
    - Attributes:
        - page_files: list of string path to every existing page
        - list_pages: list of existing page objects
    - Methods:
        - get_pages: create page record in the database for each page in page_files, if it’s not exists
        - get_page: get page object from db
        - add_page: add page record to the db
        - update_page: modify page data in the database
        - get_page_by_number: find page by name, return page object
- Converter: gets database tuples as inputs and returns corresponding objects
    - Methods:
        - convert_to_page
        - convert_to_snapshot

5.3 PDF MODULE

Creates PDF files using reportlab module. Takes Book instance and string path to PDF file as input and generate PDF file for current book utilizing published versions of pages. Module should add page number on top of images.

5.4 ASSEMBLER

Application main file. Contains two classes: Assembler and BookModel.

5.4.1 BOOK MODEL

PySide class responsible for displaing book data in QTableWidget. Methods and mostly attributes derived from PySide.

Custom attributes:

- book: to hold Bool class instance
- header: to define table header, Number, Published, Sent, Description

Custom methods:

- data: display necessary data in table
- set_data: save user description of each page to the database

5.4.2 ASSEMBLER

Main application class, entry point. Controls application data and UI.

Attributes:

- settings: settings module instance
- sql_file_path: path to the database file
- project_root: string path, root for current project
- versioned_pages: string path to pages
- final_pages: string path to layout pages
- book: book class instance
- book_model: BookModel class instance, data model for QTableView widget

Methods:

- init_ui: create Book and BookModel objects, read data from the database, populate data to UI
- set_project: define current project, record current project to settings.json
- help: show documentation html file in web browser
- show_page: display image of page selected in QTableView in UI. Also handles [+] and [-] functions to increase/decrease current version of page.
- publish_page: publish current version of selected page
- sent_published: copy published page versions to “to_layout” copy and record this data to the database
- generate_pdf: run procedure of creating PDF

6. TEST PLAN

6.1. OVERVIEW

To ensure the quality, reliability, and correctness of The Book Assembler application, a comprehensive testing strategy will be employed. Testing activities will encompass various levels of the system, from individual components to the entire application. This chapter outlines the testing strategy, including the types of tests, testing techniques, and test case examples.

The testing responsibilities will be assigned to the following roles:

- Developer: Perform testing during the development process.
- User (Book Author): Provides test project data (example project with several pages). Test delivered application on test and other projects. Provides feedback on the application.

6.2 TEST UNITS

Test Case Name: 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.

Test Case Name: 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.

Test Case Name: Add pages to a new 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.

Test Case Name: 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

Test Case Name: Add new pages to an existing project.
Test Case Description: User copy new pages from test project (provided by client) to a current 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 updates list of pages with new pages in UI

Test Case Name: 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

Test Case Name: 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.

Test Case Name: 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 Case Name: 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.

Test Case Name: 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.

Test Case Name: 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.