**Phase I – Sortie Logger**

**Zachary Hager, Charles Kimmel, and Matthew White**

**Section 6381**

**Group 4**

**20 September 2022**

**Revision History**

|  |  |  |
| --- | --- | --- |
| **Name** | **Date** | **Description** |
| Zachary Hager | 8/22/2022 | Initial GUI with drop downs and text boxes |
| Matthew White | 8/29/2022 | Refinements to critical components. Add data storing methods |
| Zachary Hager | 9/9/2022 | Migrated to tab-based interface & object “Sortie” coordination between 3 tabs (user)/1 class (code) |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**User’s Guide**

Package: Sorties

Files (.java): Main, Window GUI, Sortie, DataHandler

1. Download Group4\_Phase1.zip folder.

2. Extract the Sorties folder.

3. (On Windows OS) right-click the extracted “Sorties” folder.

4. Click “Open Folder as *[Preferred/Installed IDE]* Project”

5. Once IDE opens, compile and run the package

6. Once GUI appears click any of the three tabs to interact with the program.

* The displayed examples were made using IntelliJ IDEA. Eclipse was also used in development. This Guide is not all encompassing. This is merely amongst the easiest ways to run our program across any Windows PC, and IDE.

**Phase I Milestone**

Our milestone for Phase I of our Sortie Logger was to have a running GUI with interactable fields and tabs. We also aimed to begin to build towards storing and handling data which will come in later phases. Currently we have built the classes for storage and handling; however, they do not yet function. Overall, we have met (and in some ways incrementally exceeded) our goals for the first Phase with the body of the GUI and editable fields.

**Schedule**

Our project is currently on schedule as we have met the criteria for Phase I chiefly of which was having a working program with subclasses, we can build upon for data handling. For Phase II we will be expanding our exception handling and error catching. Additionally, we will be implementing our database which will direct saved “sorties” into our program. The brunt of what you are currently interacting with is only 2 of the 4 class, Main.java and WindowGUI.java. We will be further expanding our Sortie.java class for the Sortie object, and DataHandler.java class for storing added data.

**Reevaluations/changes:**

The primary change from the initial proposal so far was to have multiple tables handling the “Add or View” component. Initially we proposed to ask the user what they wanted to do upfront and then proceed to that part of the program. This however would require restarting the program if the user wanted to go back and forth, thus tabs were implemented for the user to switch between functions as they wish.

Our biggest hurdle for the next phase will be how we handle the stored data and how we store it. Our aim is to store it off the components of the user’s inputs in the “Add” tab. As of right now we will use the “Line Number” and “Scheduled Take-off” date as the file name as no Line Number can be used more than once the same date (i.e. Line 151 on September 18, 2022 would be: 15122261.txt). Even if the user duplicates lines it would just overwrite the file (for now) and we may integrate overwrite warnings in the future.

Going into our next Phase our main focus will be exception handling. Our biggest exception to deal with will be how to handle time deviations of over 30 minutes (i.e. If the “actual time” is 30 minutes or more off of the “scheduled time”). We have two paths forward in changing the “Cargo Description” field to a “Deviation” field, and requiring the user to input a description, or further developing the “Finalize Times” tab which is in the program right now as a potential to expand on.

**Current State as compared to Test Plan**

Below is our Test Table for our Program that was created back in Week 2. **Note** That the “PASS, PARTIAL, and FAIL” are not meant to be taken as the efficacy of the program. It is used to illustrate how Phase I lines up with where we want the program to be in the end. In a sense you can see “PASS” as the program currently meeting our objective, “PARTIAL” as meeting some but not all objectives, and “FAIL” as not meeting any objective.

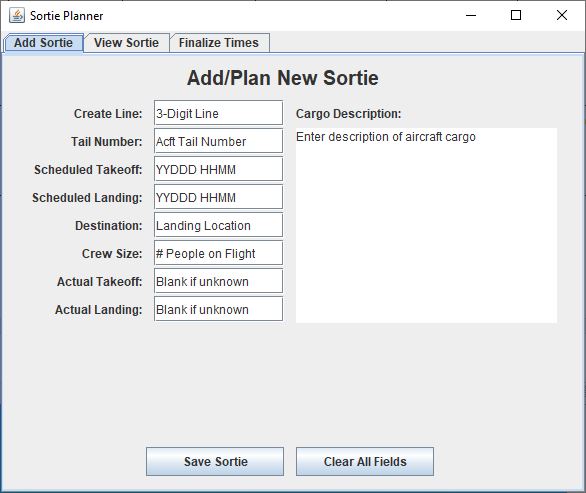
**Test Table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case # | Test Name | Test Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Start Page | Program opens with two options to proceed to adding a sortie, or retrieving a record. | Test page is present with 2 options | Test page is present with 2 options | **PASS** |
| 2 | Add Sortie | After selecting the ‘Add’ button a page to fill the details of a sortie appear. | Page to add a sortie appears. | Page to add a sortie appears. | **PASS** |
| 3 | Create Line | Acting as the sortie’s axiom, the user gives the sortie a 3-digit designation | Program accepts 3-digit line number. | Program accepts any string entry. | **PARTIAL** |
| 4 | Catching Line error | Makes the user reenter a line number if a character other than a number is entered | Prompt appears notifying the user of their error, user retries. | Does not catch any entry error by the user. | **FAIL** |
| 5 | Tail Number | Drop down appears requesting the user select a preloaded aircraft tail number. | User selects tail number, and program proceeds. | Shifted to manual entry/ Program accepts any string entry. | **PARTIAL** |
| 6 | Tail number error | Instructs user to select a tail number (aircraft) if the try to proceed without doing so. | Prompt appears notifying the user of their error, user retries. | Does not catch any entry error by the user. | **FAIL** |
| 7 | Scheduled take-off | User enters scheduled take-off date and time in Julian date notation and military time. (YYDDD HHMM) | Program accepts users’ entry. | Program accepts any string entry. | **PARTIAL** |
| 8 | Scheduled land time | User enters scheduled land date and time in Julian date notation and military time. (YYDDD HHMM) | Program accepts users’ entry. | Program accepts any string entry. | **PARTIAL** |
| 9 | Actual take-off time | User enters actual take-off date and time in Julian date notation and military time. (YYDDD HHMM) | Program accepts users’ entry. | Program accepts any string entry. | **PARTIAL** |
| 10 | Actual land time | User enters actual land date and time in Julian date notation and military time. (YYDDD HHMM) | Program accepts users’ entry. | Program accepts any string entry. | **PARTIAL** |
| 11 | Time deviation over 30 minutes | Prompts user to enter a short explanation on why the aircraft took off late. | User enter string, program accepts string | User enter string, program accepts string/user not prompted | **PARTIAL** |
| 12 | Time deviation under 30 minutes | Prompts user to enter a short explanation on why the aircraft took off early. | User enter string, program accepts string | User enter string, program accepts string/ user not prompted | **PARTIAL** |
| 13 | Date or time annotation error | User enters the date or time in the wrong format, or with wrong characters. | Prompt appears notifying the user of their error, user retries. | Prompts do not appear yet annotating errors | **FAIL** |
| 14 | Save added sortie | User saves all previous information entered. | Sortie data is saved to its own file in the program/data base. | Database not built in yet | **FAIL** |
| 15 | Data check before saving | User erroneously attempts to save data while one of the boxes is not filled. | Prompt appears notifying the user of their error, user retries without clearing the other fields. | Prompts do not appear yet annotating errors | **FAIL** |
| 16 | Retrieve Data | After selecting the ‘View Sortie’ button a page appears to retrieve sortie records. | Page appears to fill information to retrieve records. | Page appears to fill information to retrieve records. | **PASS** |
| 17 | Open archived data | User opens archived files. | User view’s record of previously added files. | Database not built in yet | **FAIL** |

**Results:**

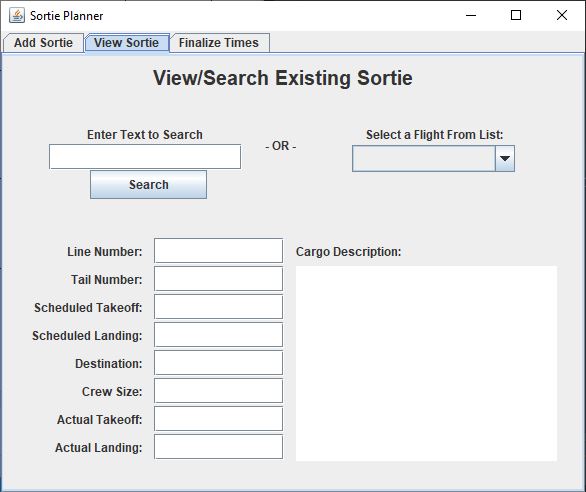
**WindowGUI.java**

* **Add Sortie:**

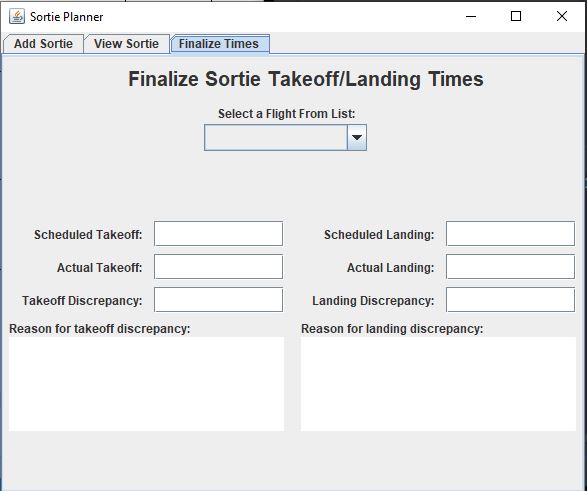


Program opens automatically to the add sortie tab. This was a change from the original design which requested the user to “Add or View” up front and then proceed. This approach is a bit more optimal as it allows the user to navigate as the program to each tab as the program is running instead of restarting the program to add or view sorties.

* **View Sortie:**



* **Finalize Times**



Experimental tab to better handle time deviations. If this falls short the “Cargo Description” in the other two tabs will be changed to “Deviations” and the user can enter the reason for the deviation in there.