

G5BUID Coursework 2

An Interface for Airport Self **Check-in**

User Guide and Implementation Notes

Sergei Golubev, sxx05u

University of Nottingham
School of Computer Science and IT

Touch me  to start

NOTE: As the second coursework is based on the previous one, coursework 1 in its specification contains all the relevant information on why this or that design or function solution is developed. Therefore, it could be useful to refer to CW1 within this coursework as well. For this reason CW1 file is included in the ZIP archive as well.

Contents

1.	User Guide	3
1.1	What prototype does?	3
1.2	How to run?.....	3
1.3	How to interact?.....	3
2.	Implementation Notes	4
2.1	Code Structure	4
2.2	Design: Graphics, Java / Swing	6
3.	Evaluation Plan	8
3.1	Usability Factors	8
3.2	Evaluation Processes.....	8
	<i>References</i>	9
	<i>Appendix</i>	10

1. User Guide

1.1 What prototype does?

The presented prototype version of the graphical user interface (GUI) is the working demonstration of the interface designed for installation on self check-in kiosks at airports. The implementation is intended to give you an opportunity for product evaluation to make sure that the interface suits all your needs before the development of the final version.

1.2 How to run?

To run the GUI the following steps have to be completed:

Under Windows: Run `AirportGUI.bat` file.

Alternatively, select *Run* from *Start Menu* and in the opened window type in `cmd` or `command`. In the command prompt change to the directory with JAR file using `cd DOS-command`. Run the JAR-package having typed `java -jar AirportGUI.jar`

Under UNIX: In the shell, type `java -jar AirportGUI.jar`

After a few seconds the interface window should appear ready to use.

NB! Present installation of Java Runtime Environment (RE) is essential for launching the interface. The latest version of Java RE can be obtained by the link: <http://www.java.com/getjava/>. The program is already compiled with Java compiler version *1.5.0_06*. The program JAR-package contains all the class and image files needed to run the program.

1.3 How to interact?

Under the circumstances that no touch-screen, proposed in the hardware requirements, was available during the prototype development stage, the mouse indirect pointing device is chosen as a suitable alternative for user's interaction with the interface.

The use of the prototype is quite simple. Use your mouse to press buttons on the GUI screen:

- Top part (the header) of the screen is for language choice and shows local time.
- Central part is the area of the "main actions": forms and questions you need to complete.
- Bottom (the footer) part will help you to navigate through the screens.

Refer to the *GUI Overall Layout* illustration in section [2.2](#) of this document.

2. Implementation Notes

2.1 Code Structure

For easiness of understanding and to aid possible modifications in the future the source code is well commented, including thorough explanations on the purpose of this or that part of the program, recommendations for programmers/developers on further implementation of some additional check-in program features and a special legend of the abbreviations used within the code. Thus, all the information relevant to this section is clearly written in comments.

The program structure and principles of operation are as follows:

We start our GUI with the initialisation of a new `AirportGUI` object in the main method. Under this, within the default class constructor, all the panes are created, their layouts and other properties are set, and the calls for creating the title page (including header, main contents pane and footer) and setting up the frame are made.

As header and footer are static and do not change every screen, there is no need to spend resources on them and redraw them with the call of every page. Thus, they are coded in the separate methods.

Main methods

Description of the program's main methods:

<code>createPageX()</code>	constructs the page: sets up the inner panes for page X, constructs components (buttons, images, fields) for these panes.
<code>pxElementOrAction()</code>	is either processing data entered by the user on page number X (and giving the respective response) and/or constructing page's specific components, which can be shared by other pages: such as numeric pad used on both authentication pages.
<code>headerFlags()</code> <code>headerClock()</code>	both methods are in charge of generating a header – the language bar and a clock
<code>createNavBar()</code>	constructs a footer with buttons for navigation
<code>actionPerformed()</code>	assigns an action for all the buttons, being triggered when this or that button is pressed.
<code>createGUI()</code>	sets up the frame, look-and-feel of a components and actually displays the window on the screen

Navigation

Navigation between the screens is accomplished by: 1) removing all the contents of the central pane, 2) composing new components to be added and 3) repainting the page with these components. For this purpose there is a content pane that is a container for all other panes, so that it is easy to clear the contents of the whole screen at once for navigation.

When navigating between the screens (pages), every time the current page number is stored in a variable. Based on this current position, we make use of the page navigation without assigning previous and next page for each page. Instead we just get to the previous/next page by incrementing/decrementing the current page number value and calling `goToPage ()` that will redirect us to the required page.

If the user wants to revise or change his choices and navigates between screens back and forward, he can clearly see his choices made (on completed pages) within the current session for the reason that these choices are saved.

Data Storage/Exchange

All the data, user's choice on the screens, is saved into variables for its further upload to airport's database. As the task is about the GUI design, some of the data was emulated: for example, array with information on flights and array of passengers were already created within the program; whereas in "real world" these need to be fetched from the database. Nevertheless, in general, it is not our task to deal with the database system when designing the interface.

Miscellaneous

HTML mark-up was used for more convenient and flexible text formatting when creating labels: in case there is a bold word in the label, there is no need to create separate `JLabels`, but just using HTML for this purpose.

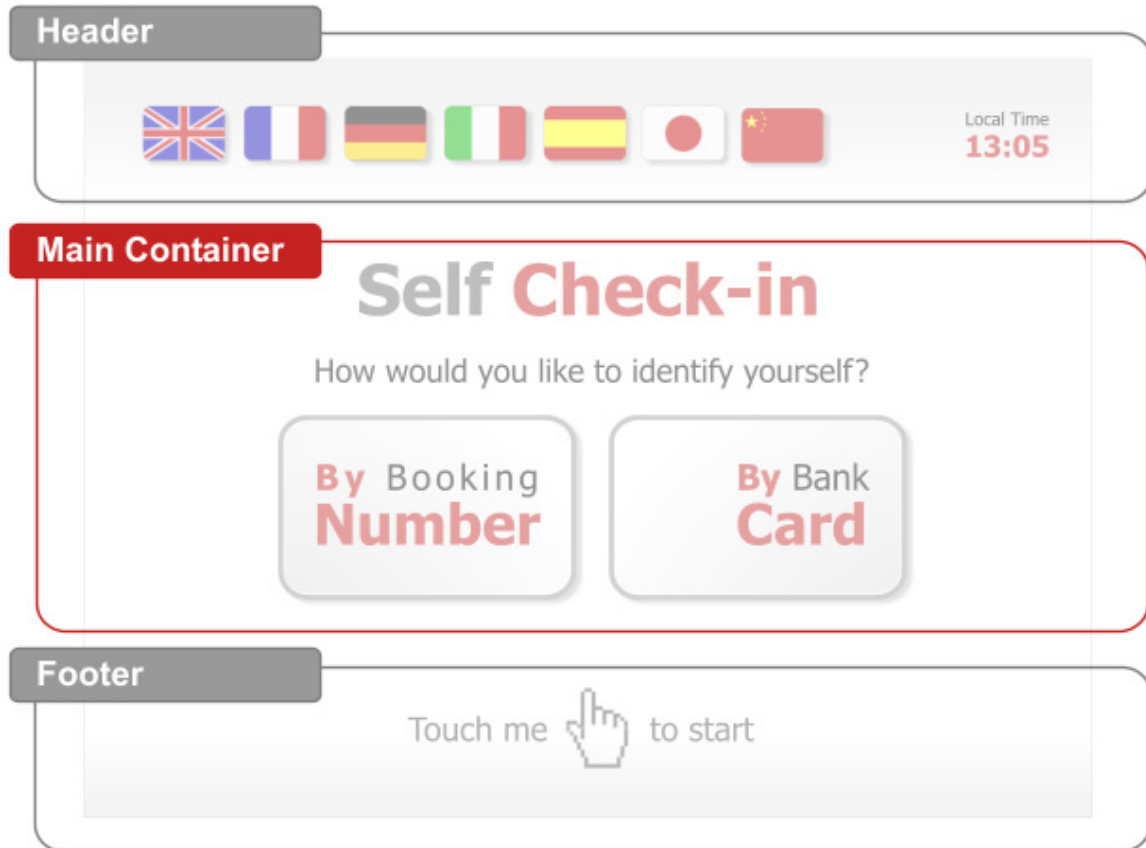
Proper checking if the section was completed and validation of the field (some number/text pattern specified for fill in, e.g. in our case – `MaskFormatter` class) was introduced to the GUI.

For testing purposes standard output is used: all actions/choices of a user are automatically printed in the console.

Many other useful functions/solutions were implemented. As it was already mentioned, the code is commented in details and it is also recommended to look through the code for the comments. On the whole, the implementation of GUI, as it was conceived in the specification, was fully completed. The solution met all the requirements stated in the specification.

2.2 Design: Graphics, Java / Swing

The overall GUI layout consists of a header (language bar), central pane (main content) and footer (start screen caption, navigation bar). Therefore, BorderLayout with its suitable structure (PAGE_START, CENTER, PAGE_END correspondingly) was chosen for the main content JPanel.



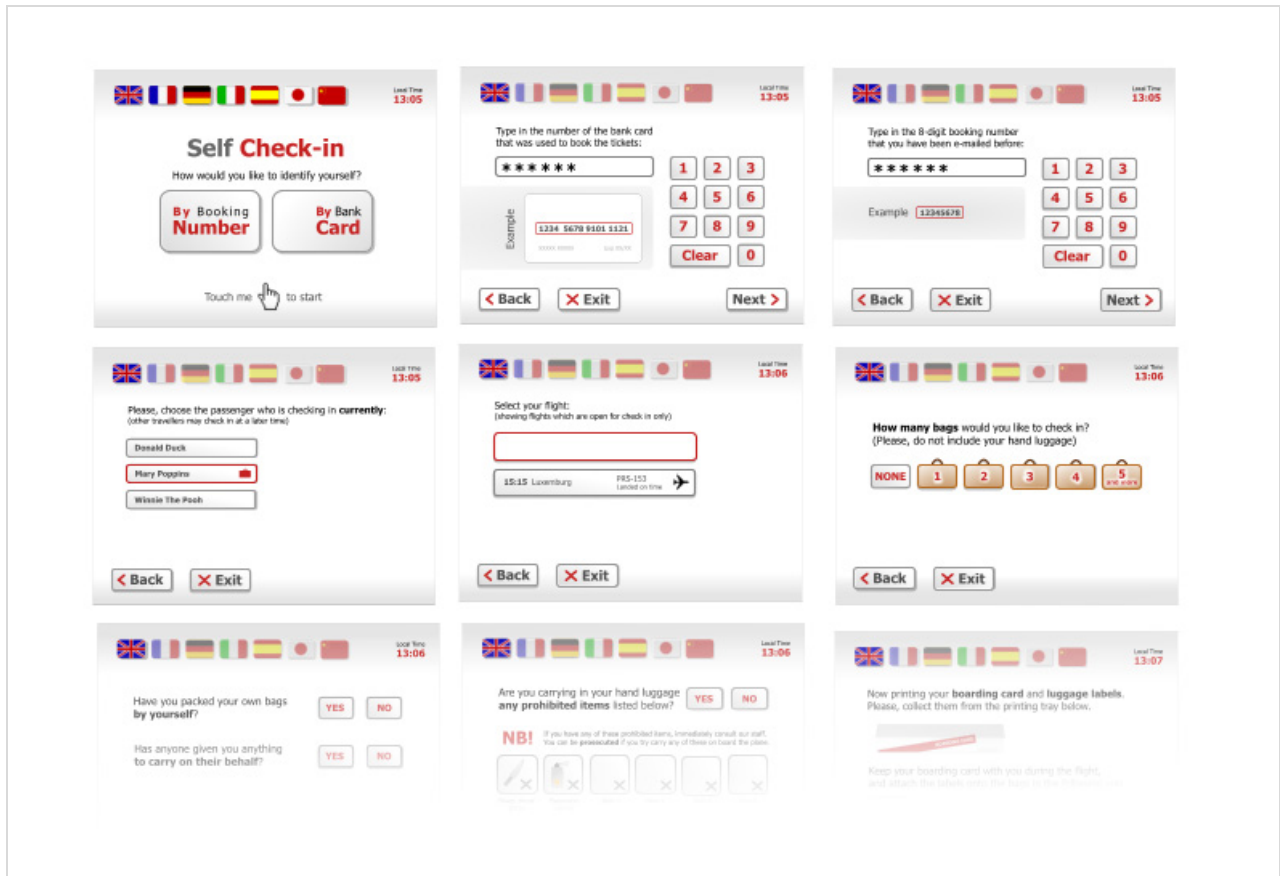
GUI Overall Layout

To make nice look and feel of the interface a set of graphics (background gradients, illustrations, and icons) was scrupulously designed. Combination of icons for the customised buttons was designed in their three possible states: initial, on-pressed and greyed-out (were made 60% transparent to highlight potential selection of other button that was activated).



Buttons and their states

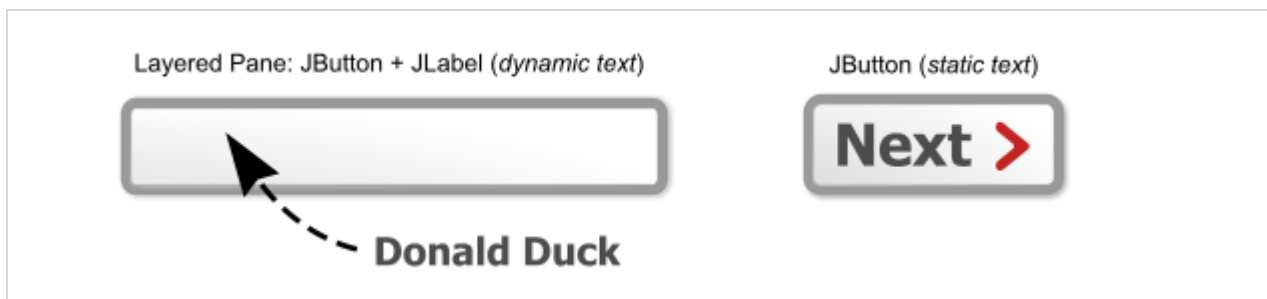
All the screens with their elements were preliminarily drawn using Xara Xtreme vector-based graphics software. See the [appendix](#) for the screenshots of all the GUI screens.



“Graphics workshop” in Xara Xtreme

Positions of navigation buttons, as a task related items, are not changed for interface consistency and easiness of an access. This is to avoid such situations when, for example, the position of "Back" button that was on the same place within five screens and suddenly on the next screen moves to another that deceives a user and his memory.

Use of layered panes was made when constructing passenger/flight selection buttons. In this case we need to place text on top of the customised button, as we deal with the dynamic text, i.e. the one that changes, in comparison with navigation buttons that already have text placed on them in the graphical editor.



Buttons creation: dynamic and static text

3. Evaluation Plan

3.1 Usability Factors

User – the best evaluator. We are aiming at the user’s satisfaction in the use of our interface. The key usability factors to be evaluated are:

- **Functionality:** what level of functional capabilities GUI provides, see if it suits user’s needs
- **Easiness:** it is clear how to use the interface within the quick time, what time is needed for learning the interface and getting used to its features
- **Layout:** organisation of the information in the way it is easily accessed and “absorbed” by the user. For quicker and more logical access there is a need to group task related items, place the frequently used elements focused on the screen’s most central area
- **Response Time (Performance):** how quick the interface works, response times (mostly relies on the hardware, thus it is good to run compatibility test on different systems to see which configuration is more appropriate)
- **Feedback:** the user is informed of any problems might occur with the use of message windows, status bars, time indicators generated by GUI
- **Maintainability:** set of features that allow facilitating and secondary type of stakeholders to easily maintain the system: update/upgrade it (includes data input), provide technical support; this also includes the easy access for the developers to the code and its flexible modification.

3.2 Evaluation Processes

It is our priority to distribute the system for evaluation over all ranges of users: primary, secondary, tertiary and facilitating types of stakeholders.

Capturing time user spends on every screen will be very useful technique for finding interface “bottlenecks”, where user could be embarrassed. This will allow us to see which screen takes longer time to complete for its further improvement, as we aim to make check-in process quicker and easier.

It would be also of an advantage to run several focus group discussion meetings within different age groups for evaluation of our software comparing to users’ needs. In order to organise cultural probes it is great to find an opportunity to develop primitive demonstrational interface simulation and distribute it over the Web to explore user’s experience and get a valuable feedback from them, of course having offered some profits to “testers” for taking part in such evaluation “experiments”, e.g. discounts for the flights (need to be negotiated with airport executives at first).

Additionally, if there is such an opportunity, experiments with camera catching user’s eyes movement could be really valuable. This allows to see which parts of the screen the user pays attention to at first; try to concentrate major interface elements in that part.

It is important to get the feedback from all of the users to improve system’s usability. In this case, questionnaires filled in by interface testers might be a valuable source of a feedback.

References

The list of the sources that after thorough research had an influence on and helped in my work. Detailed references can also be found in the source code comments. They are marked with *REF*.

Programming in Java / Swing

resources, which helped in GUI coding, learning on how to use Java Swing

- G52UID User Interface Design course lecture notes by Prof. S. Benford and Dr. B. Koleva
<http://www.crg.cs.nott.ac.uk/~sdb/uid/index.html>
- The Java Tutorials (The Swing Tutorial) on Sun Microsystems website
<http://java.sun.com/docs/books/tutorial/uiswing/>
- Roedy Green's Java & Internet Glossary
<http://mindprod.com/jgloss/jgloss.html>

Graphics design

software used for graphics design

- Xara Xtreme illustration and drawing software
<http://www.xara.com/products/xtreme/>

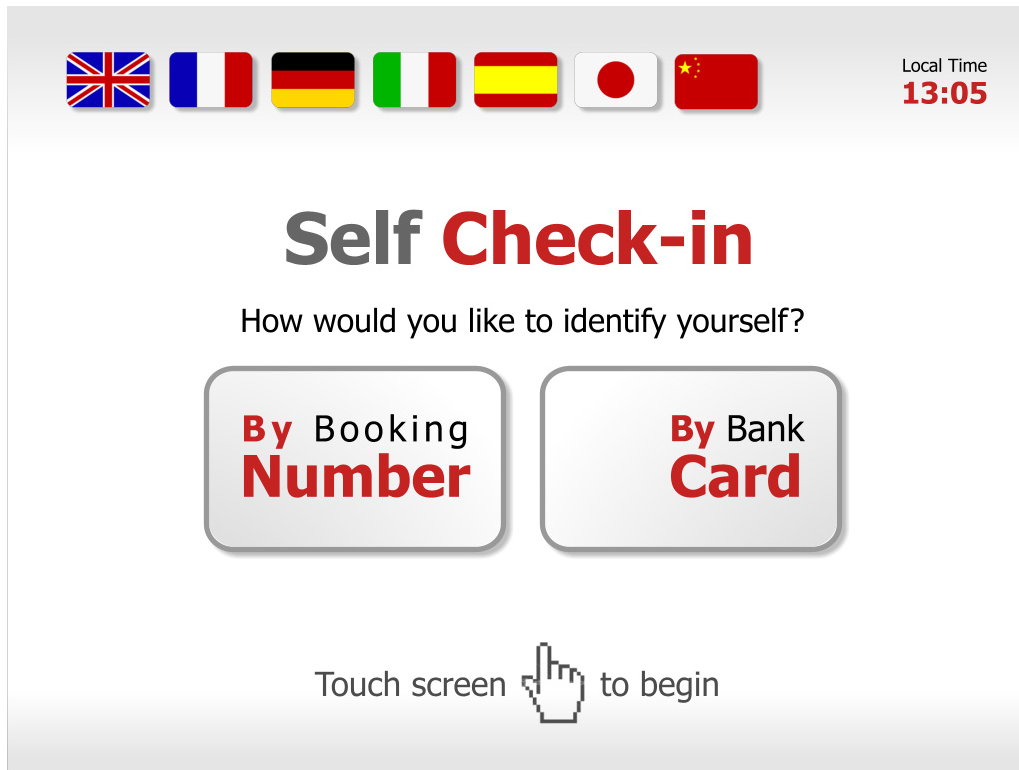
GUI specification and preliminary design (draft sketches)

- EasyJet self-check-in kiosks (at Nottingham East Midlands and Geneva):
<http://easyjet.com/EN/Flying/>
Influenced: practical approach to the system's development, check-in process stages, opportunity for multiple users' check-in
- IBM Self-service kiosk solution for airlines and airports
<http://www-03.ibm.com/industries/travel/doc/content/solution/184451106.html>
Hardware design issues
- Department for Transport (UK)
http://www.dft.gov.uk/stellent/groups/dft_about/documents/page/dft_about_612280.hcsp
Airport security issues
- Exeter Airport
http://www.exeter-airport.co.uk/plugins/print_version/print_preview.php?id=195
Check-in advice, airport procedures
- Author's own experience in the field of design, as a web-designer
<http://www.sgolubev.com/portfolio>

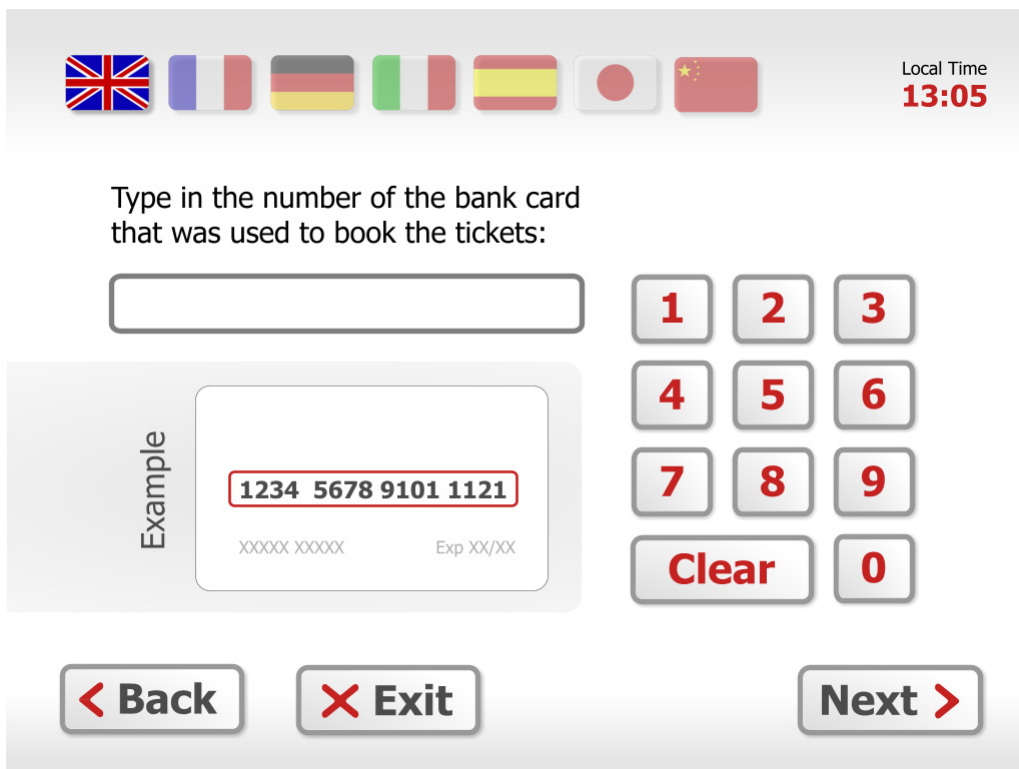
All the images, sketches, icons and charts were drawn by the author.

Appendix


Below are the screenshots of the GUI screens in the same sequence as when running the program:



Page 0 — Start screen



Page 1 — Passenger's authentication by bank card

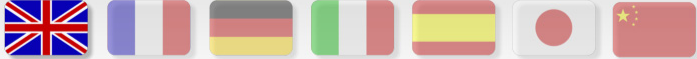
 Local Time **13:05**

Type in the 8-digit booking number that you have been e-mailed before:


Example

1	2	3
4	5	6
7	8	9
Clear	0	


Page 2(1) — Passenger’s authentication by booking reference number

 Local Time **13:05**



Please, choose the passenger who is checking in **currently**:
(other travellers may check in at a later time)



Page 2(2) — Current passenger selection


Local Time
13:06

Select your flight:
(showing flights which are open for check in only)

14:15 Paris	PRS-153 Boarding started 
15:15 Luxemburg	PRS-153 Landed on time 

Page 3 — Current flight's destination selection, informing of current flight's status.


Local Time
13:06

How many bags would you like to check in?
(Please, do not include your hand luggage)

Page 4 — Luggage enquiries



Local Time
13:06

Have you packed your own bags
by yourself?

YES

NO

Has anyone given you anything
to carry on their behalf?

YES

NO

< Back

✗ Exit

Page 5 — Security related questions (on luggage packing)



Local Time
13:06

Are you carrying in your hand luggage
any prohibited items listed below?

YES

NO

NB! If you have any of these prohibited items, immediately consult our staff.
You can be **prosecuted** if you try carry any of these on board the plane.



Sharp, thrust
Items



Flammable
Liquids



Item 3



Item 4



Item 5



Item 6

< Back

✗ Exit

Page 6 — Security related questions (on prohibited items)



Local Time
13:07

Now printing your **boarding card** and **luggage labels**.
Please, collect them from the printing tray below.



Keep your boarding card with you during the flight,
and attach the labels onto the bags in the following way:



✗ Exit

Next >

*Page 7 — Boarding card and sticky luggage labels printing,
illustrated guidance on proper attachment of the labels*



Local Time
13:07

You may now take your bags into **Bag Drop zone**.

Please, ensure you have got your boarding card with you
and all the sticky luggage labels attached to the bags.

After you drop the bags, proceed to **GATE 3**
Your plane is on time. Boarding time **14:00**.

Thank you for flying with A2B Airlines,
We wish you a very pleasant flight!

✓ Finish

Page 8 — Final screen