Dr. Yahia Fares University of Medea

Faculty of Science

Department of Mathematics and Computer Science

3rd Year in Computer Systems

Academic Year: 2025/2026

Practical Work 1

Introduction to the Design and Implementation of HMIs

HMI Registration & Authentication Report

Student: Bouziani Alaa Eddine

Group: 2

Instructor: Practical Work Supervisor

Contents

1 Objectives

This practical assignment focuses on three complementary objectives within Human–Machine Interaction (HMI):

1. **Graphical User Interface Design**: craft intuitive registration and authentication forms that respect usability and accessibility best practices.
2. **Human–Computer Interaction Modeling**: describe user journeys, validation rules, and navigation paths through structured diagrams and state machines.
3. **Human–Computer Interaction Implementation**: realise the designed interfaces with NetBeans' Swing GUI Builder, ensuring the behaviour matches the specified models.

2 Context

The case study considers a desktop application that handles both user registration and authentication. The registration form initially presents nine interactive elements:

- Last Name (optional)
- First Name (optional)
- Username
- Email
- Phone Number
- Password
- Confirm Password
- Validate button
- Cancel button

The interface must clearly communicate validation errors after the user presses the *Validate* button, and it must reassure the user when the submission succeeds by showing a summary screen with the captured information and two controls: a final *Validate* button and a *Back* button for corrections.

Field Constraints

Field	Description and Validation Rule
Last Name	Optional. When provided, it is displayed verbatim in the confirmation screen.
First Name	Optional. When provided, it is displayed verbatim in the confirmation screen.
Username	Required if both Last Name and First Name are empty. The value must be non-empty once that condition applies.
Email	Required if the Phone Number is empty. Must follow a valid email pattern (`local@domain`).
Phone Number	Required if the Email is empty. Must respect the application's digit and length constraints.
Password	Always required. Collected securely and never echoed back in plaintext.
Confirm Password	Always required and must match the Password field exactly.
Validate	Submits the form, triggering validation and either success confirmation or targeted feedback.
Cancel	Aborts the process and returns the user to the neutral state.

Success Workflow

When every constraint is satisfied, the user advances to a confirmation view summarising all captured inputs (excluding raw passwords). The user must explicitly confirm the submission or return to the editable form via the *Back* button to adjust any field before final validation.

3 Registration Flow

This section documents each state of the registration journey. Screenshots are labelled R0–R9 and should be read together with the explanations and validation notes provided beneath every figure.

Initial State

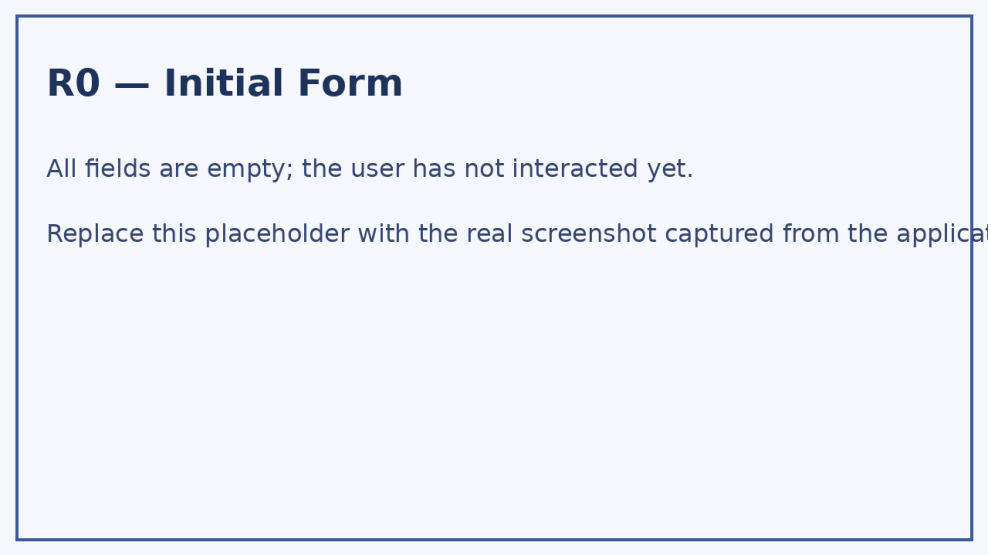

Figure 1: R0 — Registration form presented in its initial, empty state.

Explanation: The baseline interface exposes all nine components with neutral messaging. No validation feedback is visible until the user attempts to submit the form.

Validation Errors

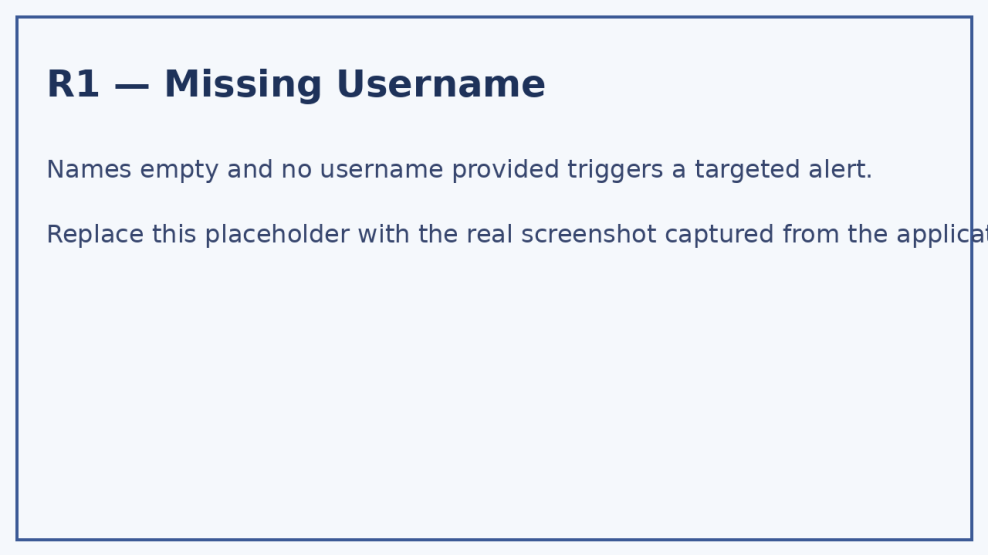

Figure 2: R1 — Missing username while both names are omitted.

Explanation: When the user provides neither last nor first name, the form demands a username to avoid anonymous account creation. A contextual message clarifies the rule.

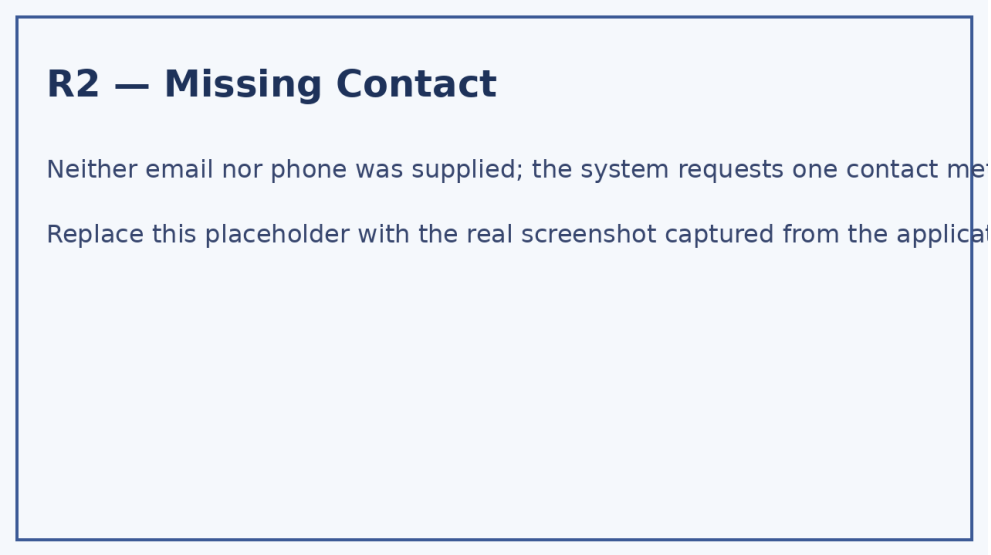

Figure 3: R2 — Missing both email and phone number.

Explanation: At least one contact channel must be captured. The feedback highlights both fields and suggests filling at least one of them.

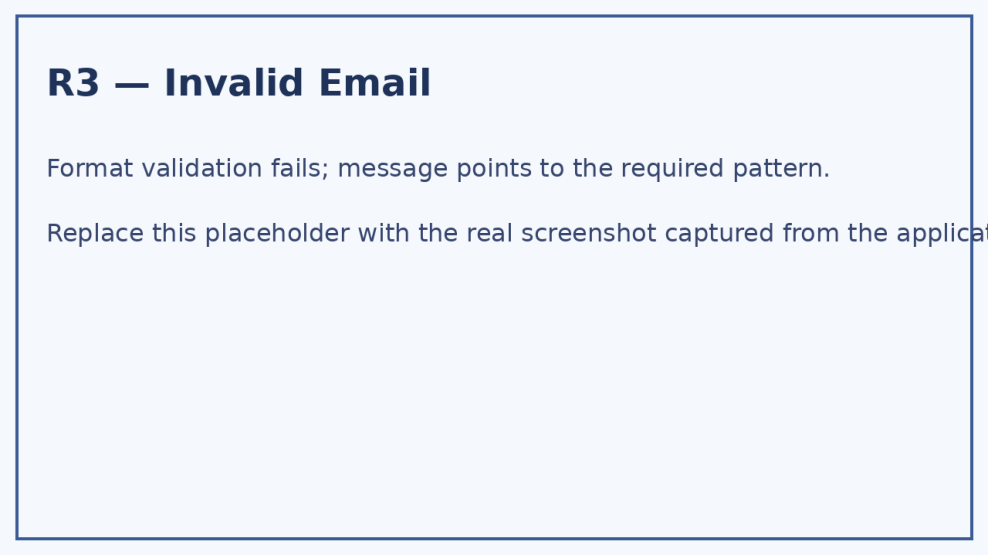

Figure 4: R3 — Invalid email format.

Explanation: The validation pattern rejects malformed addresses and informs the user about the proper format (e.g., presence of '@' and domain suffix).

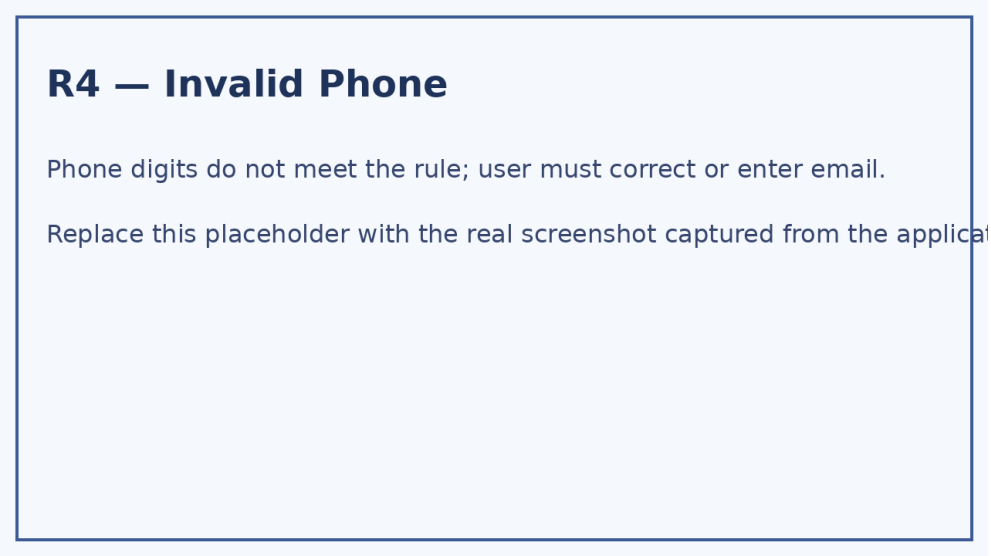

Figure 5: R4 — Invalid phone number format.

Explanation: The phone field checks for permitted digits and length. The message prompts the user to correct the number or provide an email instead.

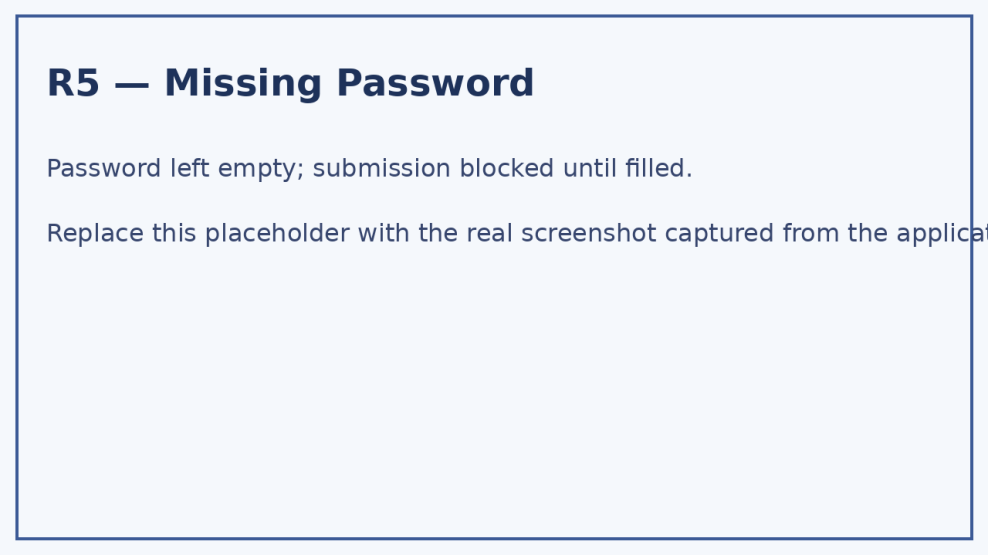

Figure 6: R5 — Password left empty.

Explanation: The password remains mandatory regardless of other inputs. The warning encourages the user to set a strong secret.

Figure 7: R6 — Confirm password left empty.

Explanation: The confirmation field ensures the user intentionally typed the password. The message directs the user to complete the secondary entry.

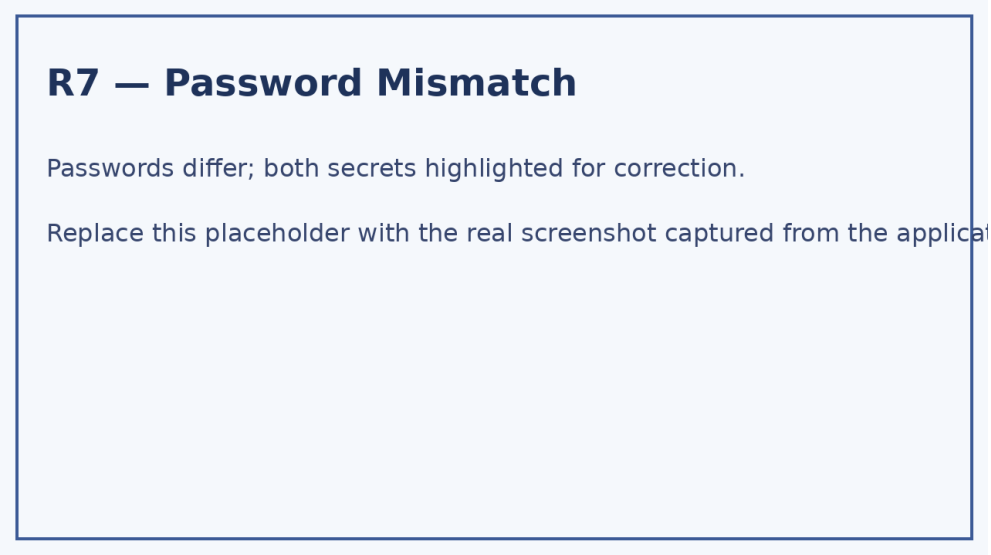

Figure 8: R7 — Confirmation does not match the password.

Explanation: Whenever the two secret fields diverge, both inputs are marked and the user is invited to retype them for consistency.

Successful Submission

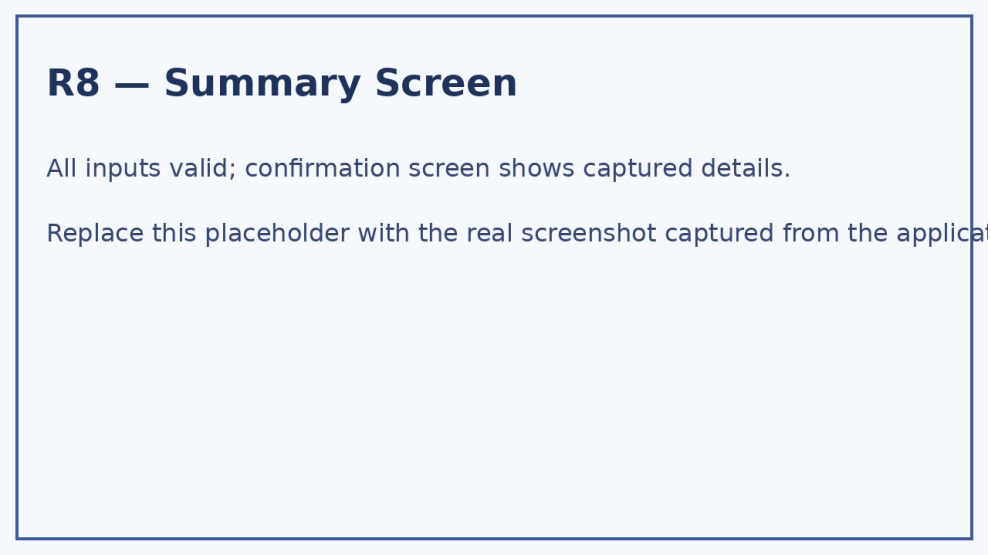

Figure 9: R8 — Summary screen confirming the captured data before final validation.

Explanation: All validated information (except the raw password) is echoed back. The user may press *Validate* to persist the account or *Back* to revise inputs.

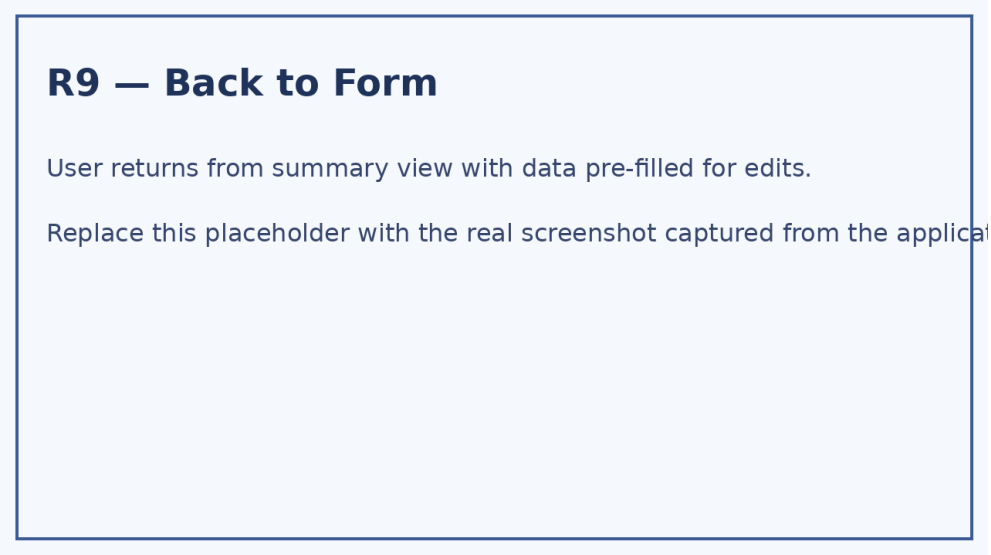

Figure 10: R9 — Returning to the editable form after pressing Back from the summary.

Explanation: The form reopens with the previous entries pre-filled so the user can fine-tune specific fields before revalidating.

4 Authentication Flow

The authentication module mirrors the registration attention to feedback and recoverability. Screenshots A0–A7 capture typical user pathways and edge cases.

Baseline and Errors

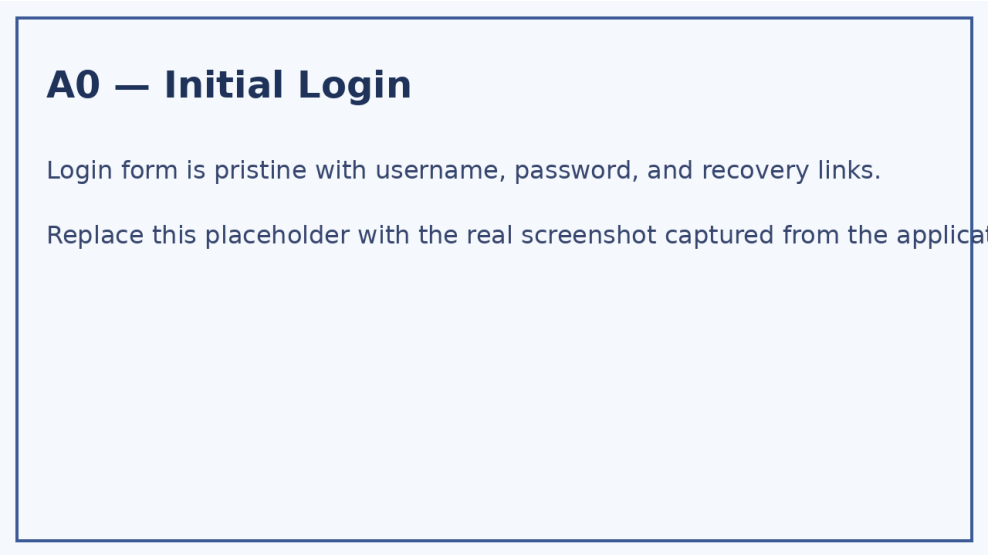

Figure 11: A0 — Initial authentication form with neutral state.

Explanation: The login UI displays username and password fields alongside action buttons and recovery shortcuts, with no error feedback yet.

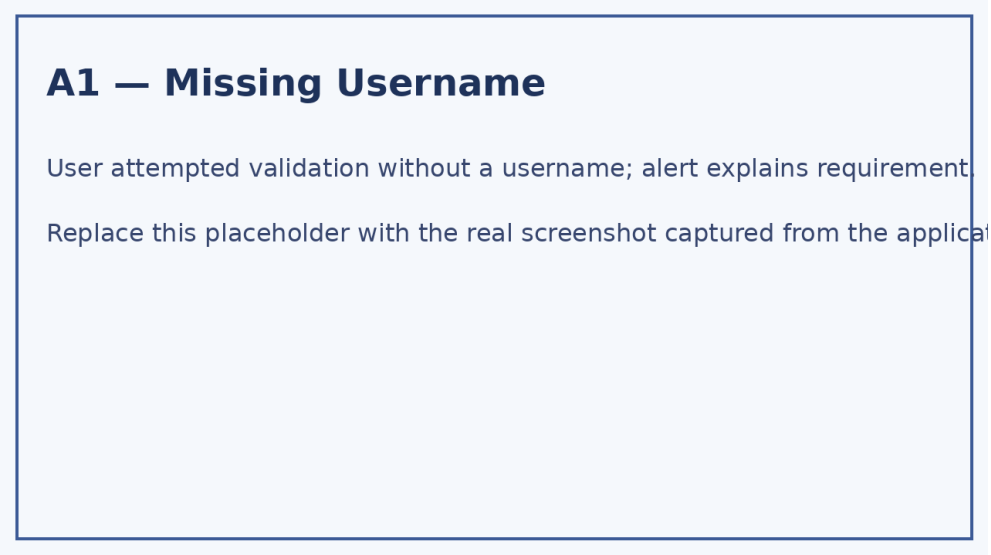

Figure 12: A1 — Username left empty.

Explanation: A contextual alert invites the user to enter their username before proceeding.

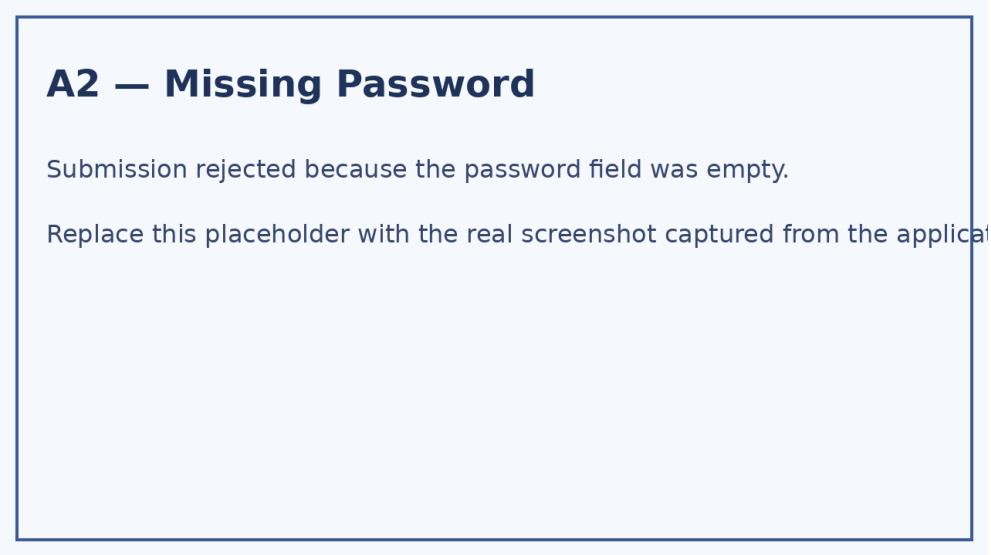

Figure 13: A2 — Password left empty.

Explanation: The password field becomes highlighted with guidance to provide the secret credential.

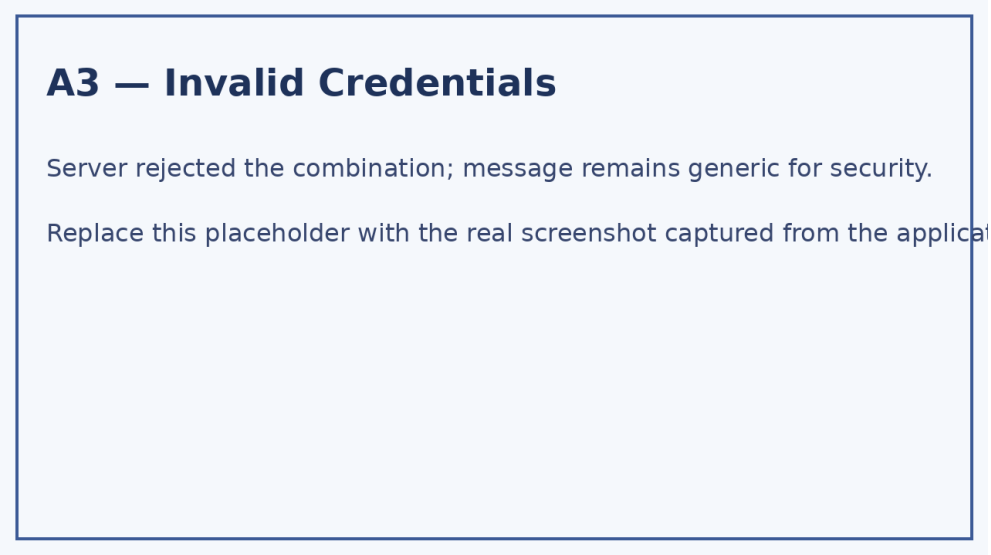

Figure 14: A3 — Invalid username or password combination.

Explanation: After submission, the form informs the user that the provided credentials do not match any account, without revealing which field was incorrect.

Recovery and Success

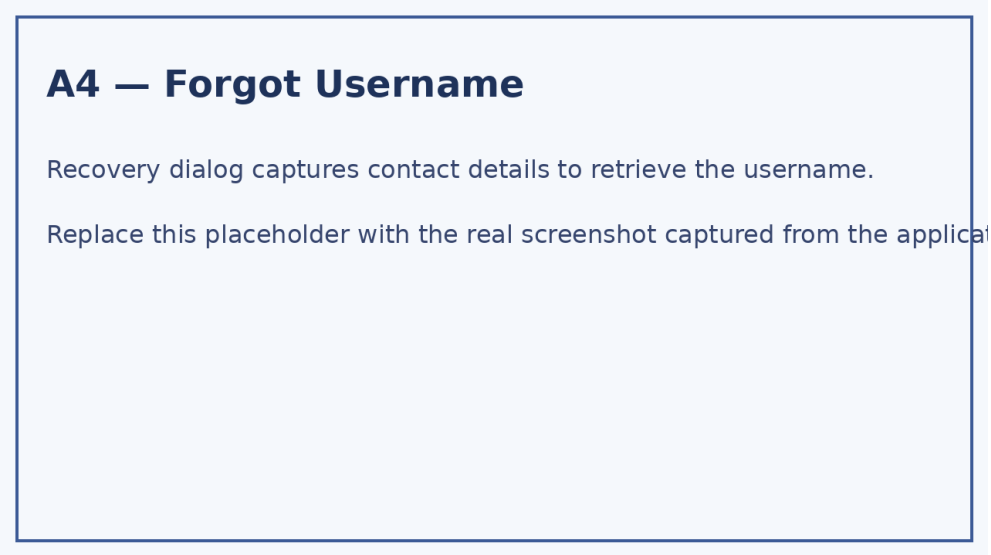

Figure 15: A4 — Forgotten username workflow.

Explanation: The recovery dialog collects an email or phone number to send username reminders, reinforcing continuity for users who forgot their identifier.

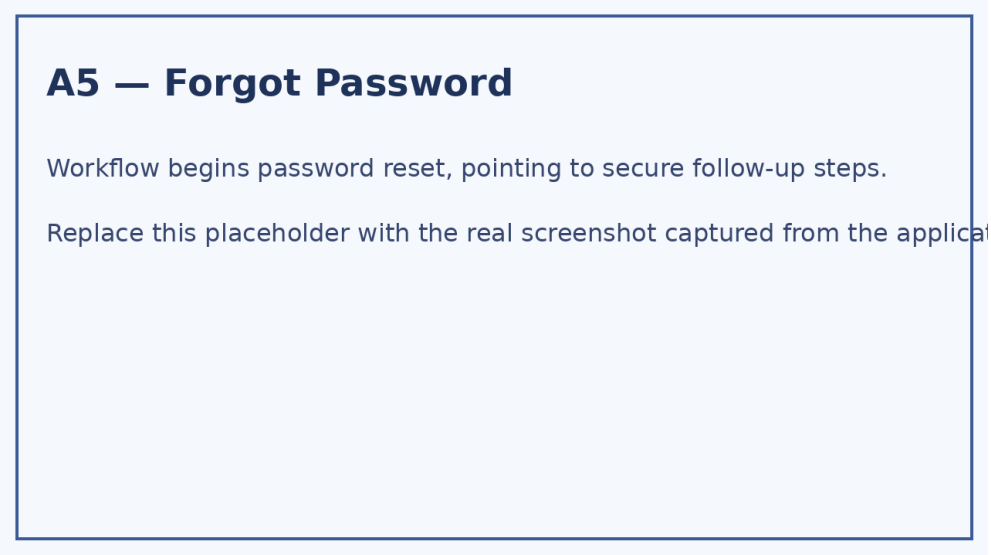

Figure 16: A5 — Forgotten password workflow.

Explanation: The process leads users through password reset instructions while maintaining a consistent look and feel with the main authentication form.

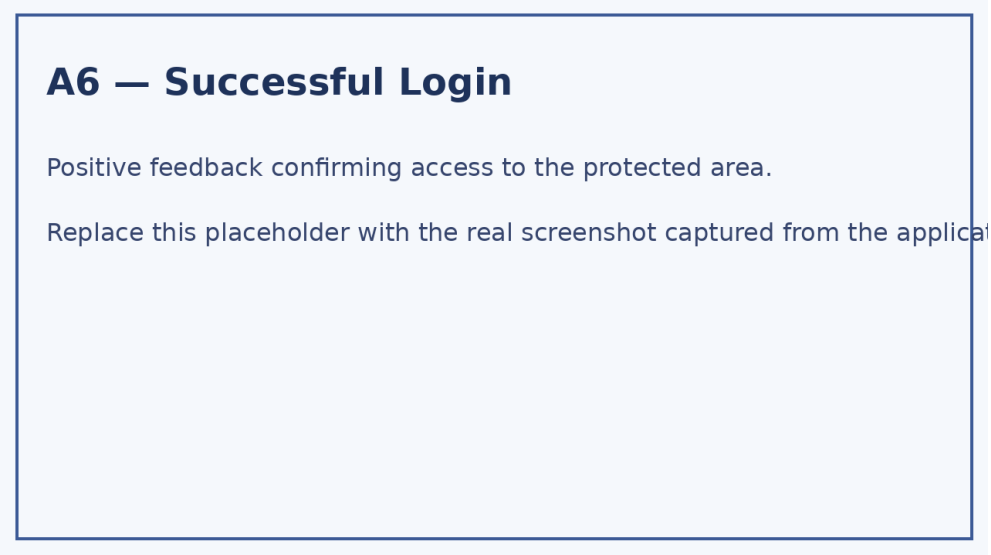

Figure 17: A6 — Successful authentication.

Explanation: Upon valid credentials, the interface confirms success and transitions to the secured area (or displays a confirmation message, as illustrated).

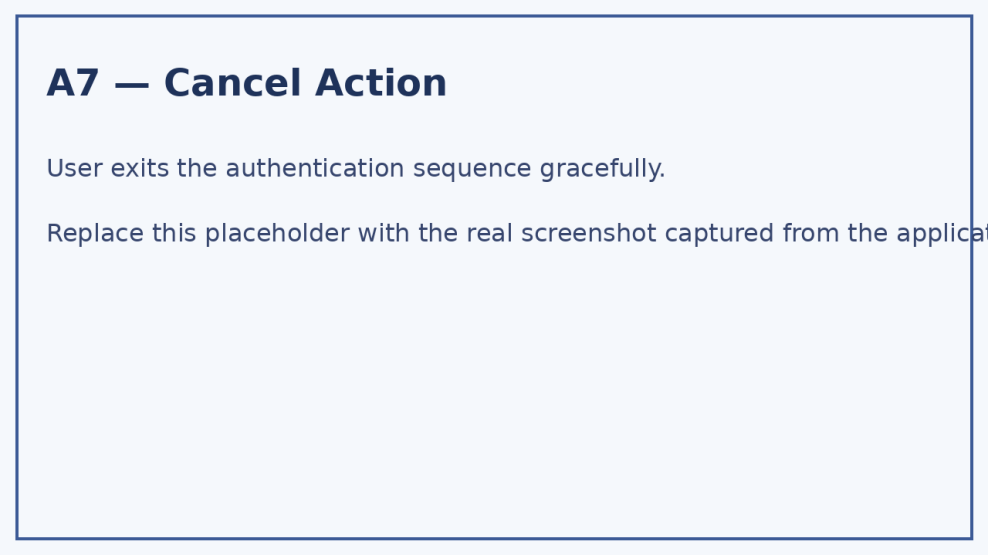

Figure 18: A7 — Cancel action returning to the neutral state.

Explanation: The cancel button aborts the attempt and closes or resets the form, ensuring users can exit gracefully without residual messages.

5 Interaction Automata

The numbered screens documented earlier map directly to states in two deterministic finite automata. Each transition is labelled with the initiating action and, where relevant, the validation outcome.

Registration Automaton

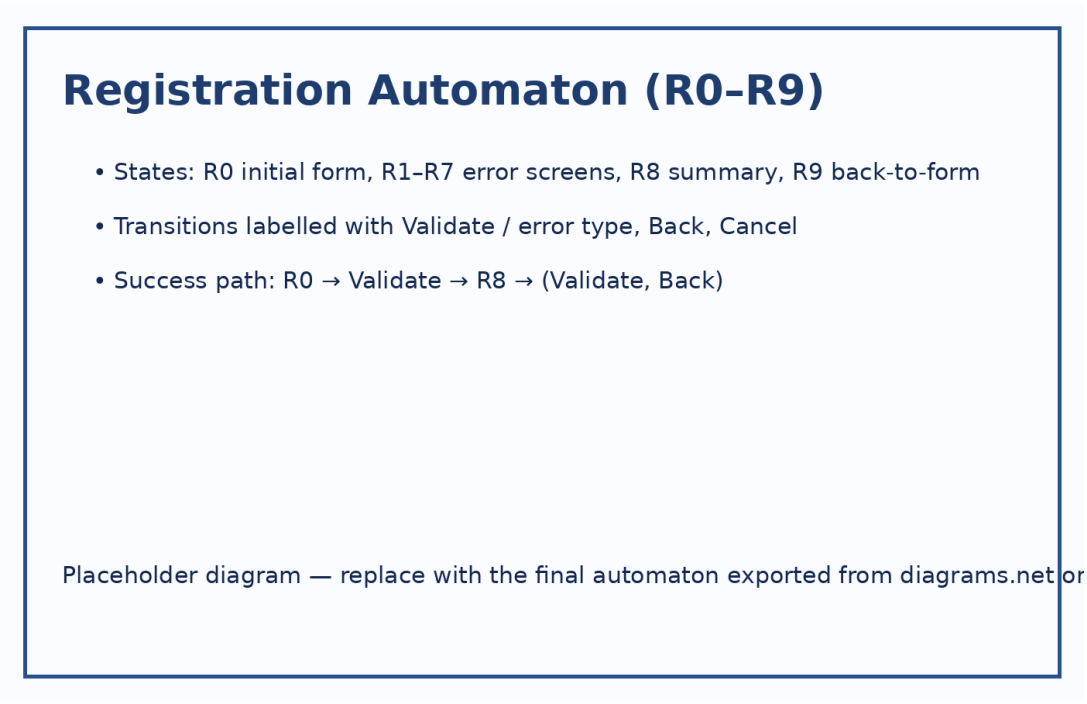

Figure 19: Automaton linking registration states R0–R9.

Explanation: The graph emphasises how every submission funnel returns to R0 when the user presses *Cancel*, while the *Validate* action either routes to an error-specific state or advances to the confirmation node R8. From R8, the user may finalise the account or traverse *Back* to R9 for edits, then re-validate.

Authentication Automaton

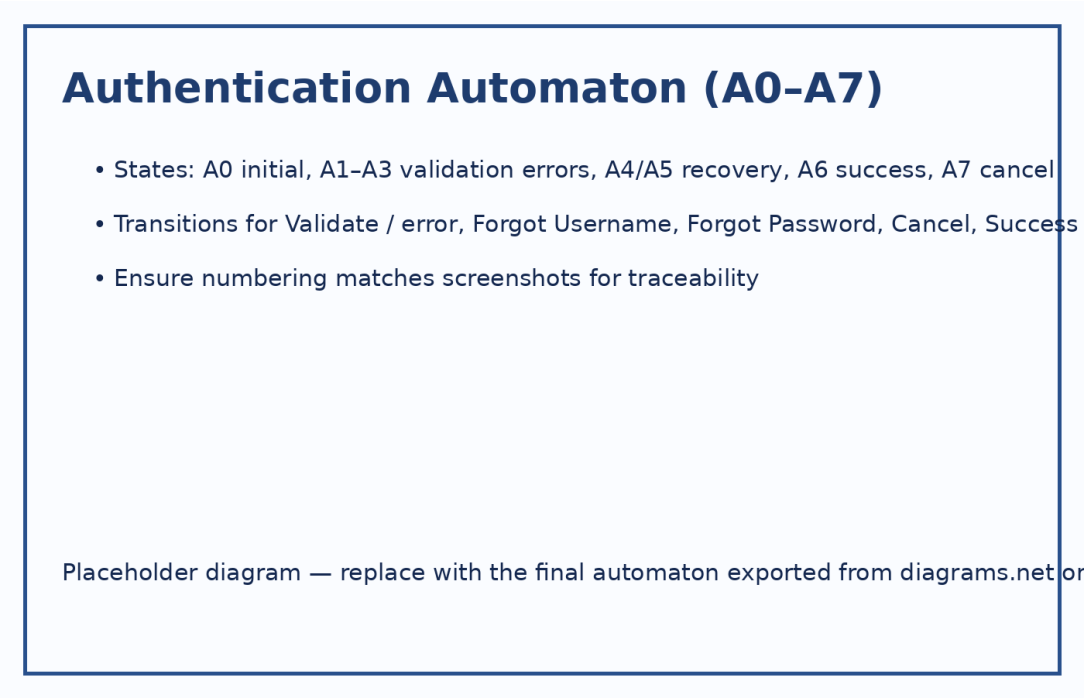

Figure 20: Automaton linking authentication states A0–A7.

Explanation: The authentication flow features clear exits for recovery scenarios and maintains a return path to A0 after cancellations or successful login completions.

6 Implementation Notes

The NetBeans Swing GUI Builder was used to instantiate the designed interfaces. Key steps:

1. Create a new Java Swing form named `LoginRegisterTabbed` and configure a tabbed pane containing the registration and authentication panels.
2. Drag-and-drop Swing components (labels, text fields, password fields, buttons) while aligning them using NetBeans' layout guides to maintain consistent spacing.
3. Attach event handlers to the *Validate*, *Cancel*, and recovery buttons. Validation logic follows the rules summarised in Section 2.
   The full implementation is available at `/home/alaa/hmi/LoginRegisterTabbed.java`.
4. Display contextual errors through coloured labels or dialog boxes. On successful validation, open the confirmation dialog (registration) or navigate to the protected view (authentication).
5. Capture the screenshots listed earlier by running the application, populating the form with test data, and using the operating system's screenshot utility. Save the resulting images into `docs/images/`.

Each screen capture should correspond to the state identifiers used in the automata to preserve traceability between design artefacts and the implementation.

7 Conclusion

The iterative combination of interface design, explicit validation feedback, and deterministic modeling ensures that end users understand both the registration and authentication processes. Linking the GUI states to automata clarifies navigation continuity, while the NetBeans implementation demonstrates feasibility. Future work may include usability testing sessions and internationalisation of messages so the application can serve a broader audience with the same interaction quality.