**TP1 – Input Validation Plan**

Team: Team <n>  
Purpose: Identify and enforce validation for every input field across the application. Validation must occur **before** processing, surface meaningful errors, and block submission until fixed.

**1) Global Validation Principles**

**• Validate on both UI and controller/model** (trust nothing from the client).

**• Fail-fast**: do not process/persist until all fields pass validation.

**• Helpful messages**: field-specific, actionable (“Email must be like name@example.com”).

**• Centralize helpers**: shared Validation utilities; reuse everywhere.

**• Security**: treat all inputs as untrusted; trim & normalize; avoid injection/XSS.

**• Accessibility**: errors linked to fields; preserve user input on failure.

**• Internationalization**: don’t over-restrict valid names/punctuation.

**2) Shared Validation Helpers (to implement)**

• boolean isNonEmpty(String s) → not null; trim().length >= 1.

• boolean maxLen(String s, int n) → after trim, length ≤ n.

• boolean matchesRegex(String s, String pattern)

• boolean isEmail(String s) → RFC 5322 subset: local@domain.tld, no spaces, domain has a dot.

• boolean isStrongPassword(String s) → ≥12 chars, upper+lower+digit+symbol, no leading/trailing spaces.

• boolean isUsername(String s) → 3–32 chars, letters/digits/underscore/dot, starts with letter, no consecutive dots.

• boolean isPhone(String s) → digits with optional +, spaces, dashes, parentheses; total 10–15 digits.

• boolean isInSet(String s, Set<String> allowed) → for dropdowns/roles/actions.

• boolean isPositiveInt(String s) → parseable int ≥ 1.

• boolean isSafeFreeText(String s) → length cap (e.g., 1000), no control chars; escape on render.

• String normalize(String s) → trim, collapse inner whitespace, Unicode NFC.

Consider a Validation.java utility or static methods on a shared Model/Utils class.

**3) Login (ViewUserLogin / ControllerUserLogin)**

**Username/Email**

• Rules: isNonEmpty, (isUsername OR isEmail), maxLen(64)

• Message: “Enter a valid username or email.”

• Enforce in: ControllerUserLogin.validateCredentials(...)

**Password**

• Rules: isNonEmpty, maxLen(128)

• Message: “Enter your password.”

• Enforce in: ControllerUserLogin.validateCredentials(...)

**4) New Account (ViewNewAccount / ControllerNewAccount)**

**First Name**

• Rules: isNonEmpty, maxLen(40), isSafeFreeText (allow hyphen/apostrophe)

• Message: “Enter your first name (max 40).”

• Enforce: ControllerNewAccount.validateAndCreate(...)

**Last Name**

• Rules: isNonEmpty, maxLen(60), isSafeFreeText

• Message: “Enter your last name (max 60).”

• Enforce: ControllerNewAccount.validateAndCreate(...)

**Username**

• Rules: isUsername, maxLen(32), **uniqueness** in Database

• Message: “Choose a username (letters, digits, \_ or .); that one is taken.”

• Enforce: controller + Database.existsUser(...)

**Email**

• Rules: isEmail, maxLen(254), **uniqueness**

• Message: “Enter a valid email; that email is already in use.”

• Enforce: controller + Database.existsEmail(...)

**Phone (optional)**

• Rules: if present → isPhone, maxLen(20)

• Message: “Enter a valid phone number or leave blank.”

• Enforce: controller

**Password**

• Rules: isStrongPassword, maxLen(128), reject common passwords, not whitespace-only

• Message: “Use at least 12 chars with upper, lower, number, and symbol.”

• Enforce: controller

**Confirm Password**

• Rules: equals(Password)

• Message: “Passwords do not match.”

• Enforce: controller

**Role selection (if applicable)**

• Rules: isInSet(allowedRoles)

• Message: “Invalid role selection.”

• Enforce: controller

**5) Update Account (ViewUserUpdate / ControllerUserUpdate)**

**First/Display Name**

• Rules: isNonEmpty, maxLen(40), isSafeFreeText

• Message: “Enter a valid first name (max 40).”

• Enforce: ControllerUserUpdate.validateAndUpdate(...)

**Last Name**

• Rules: isNonEmpty, maxLen(60), isSafeFreeText

• Message: “Enter a valid last name (max 60).”

• Enforce: controller

**Email (on change)**

• Rules: isEmail, maxLen(254), **uniqueness if changed**

• Message: “Enter a valid email; that email is already in use.”

• Enforce: controller + Database

**Phone (optional)**

• Rules: if present → isPhone, maxLen(20)

• Message: “Enter a valid phone number or leave blank.”

• Enforce: controller

**Change Password: current**

• Rules: isNonEmpty; must match stored hash

• Message: “Enter your current password.”

• Enforce: ControllerUserUpdate.changePassword(...)

**Change Password: new**

• Rules: isStrongPassword, maxLen(128), must differ from current

• Message: “Use a stronger password different from your current one.”

• Enforce: controller

**Confirm New Password**

• Rules: equals(New Password)

• Message: “Passwords do not match.”

• Enforce: controller

**6) Role Dispatch (Single/Multiple Role Dispatch)**

**Role selection**

• Rules: isInSet(rolesForUser) (prevent tampering)

• Message: “Invalid role.”

• Enforce: ControllerMultipleRoleDispatch.dispatch(...)

**7) Admin – First Admin Setup (ViewFirstAdmin / ControllerFirstAdmin)**

**Admin Username**

• Rules: isUsername, maxLen(32), **uniqueness**

• Message: “Enter a valid, unique username.”

• Enforce: ControllerFirstAdmin.initializeAdmin(...)

**Admin Email**

• Rules: isEmail, maxLen(254), **uniqueness**

• Message: “Enter a valid, unique email.”

• Enforce: controller

**Temporary Password**

• Rules: isStrongPassword, maxLen(128)

• Message: “Provide a strong temporary password.”

• Enforce: controller

**8) Admin – User & Role Management (ViewAdminHome / ViewAddRemoveRoles)**

**Lookup Username/Email**

• Rules: isNonEmpty, (isUsername OR isEmail), must exist

• Message: “User not found.”

• Enforce: ControllerAdminHome.findUser(...)

**Assign Role**

• Rules: isInSet(allowedRoles), cannot assign duplicate role

• Message: “Invalid or duplicate role.”

• Enforce: ControllerAddRemoveRoles.assignRole(...)

**Remove Role**

• Rules: isInSet(userCurrentRoles); obey policy (e.g., cannot remove final required role)

• Message: “Cannot remove the last required role.”

• Enforce: ControllerAddRemoveRoles.removeRole(...)

**9) Role 1 – Home (ViewRole1Home / ControllerRole1Home) — Project-Specific**

(Replace with your actual fields)

**• Course ID**: isPositiveInt; must exist in Database.courses

◦ Msg: “Select a valid course.”

◦ Enforce: controller

**• Notes (optional)**: isSafeFreeText, maxLen(500)

◦ Msg: “Notes are too long.”

◦ Enforce: controller

**10) Role 2 – Home (ViewRole2Home / ControllerRole2Home) — Project-Specific**

(Replace with your actual fields)

**• Student ID**: isPositiveInt; must exist in Database.users

◦ Msg: “Select a valid student.”

◦ Enforce: controller

**• Action Dropdown**: isInSet(allowedActions)

◦ Msg: “Invalid action.”

◦ Enforce: controller

**11) Security Validation & Hardening**

**• Trim & normalize** inputs; collapse whitespace.

**• Output encoding** for any user-provided text when rendering (prevents XSS).

**• Reject control chars**; restrict HTML tags in free text.

**• Passwords**: never log; hash with PBKDF2/BCrypt/Argon2, per-user salt; constant-time compare.

**• Rate limit** login; lockout or CAPTCHA after N failures.

**• Audit/log** validation failures (no sensitive data).

**• CSRF tokens** for state-changing actions (if applicable).

**12) Error Message Guidelines**

• Be specific and actionable; one visible error per field at a time.

• Highlight the field with the error; keep the user’s input intact.

• On login, don’t reveal which part failed: “Invalid credentials.”

**13) Validation Test Plan (for Screencast #1)**

**• Unit tests** for helper methods (boundaries, Unicode, invalid patterns).

**• Controller tests** for success/failure; no processing on invalid input.

**• Semi-automated checks** similar to PasswordEvaluationTestingAutomation.

**• Manual exploratory**: long inputs, copy/paste spaces, emojis, SQL-ish strings.

**14) Ownership & Traceability**

For each field above, list the exact controller method where validation occurs and the helper used. Keep this spec updated as code changes.

**Sign-Off**

• Name | ASURITE | Signature | Date (all team members)