Automatisierte Sonderfalltests (JUnit mit assert)

Sonderfall: Rechnung darf nur im aktuellen Monat und bei Status „SUBMITTED“ editierbar sein.

```
    @Test
    void testIsEditable() {
        Invoice inv = new Invoice( fileName: "file.png", Invoice.InvoiceCategory.RESTAURANT,  invoiceAmount: 10.0);
        LocalDate today = LocalDate.now();
        inv.setSubmissionDate(today);
        inv.setStatus(Invoice.InvoiceStatus.SUBMITTED);
        assertTrue(inv.isEditable(),  message: "heutige, nicht-approved Rechnung sollte editierbar sein");

        inv.setStatus(Invoice.InvoiceStatus.APPROVED);
        assertFalse(inv.isEditable(),  message: "approved darf nicht mehr editierbar sein");

        inv.setStatus(Invoice.InvoiceStatus.SUBMITTED);
        inv.setSubmissionDate(today.minusMonths( monthsToSubtract: 1));
        assertFalse(inv.isEditable(),  message: "Monat zurückliegende Rechnung darf nicht editierbar sein");
    }
```

Sonderfall: Erstattung darf keinen negativen Betrag enthalten.

```
    @Test
    void testUpdateRestaurantAmount() {
        RefundConfigDAO dao = new RefundConfigDAO();
        double newAmount = 4.0;

        boolean success = dao.updateAmount( category: "RESTAURANT", newAmount);
        assertTrue(success,  message: "Administrator should be able to update reimbursement amount");

        double updatedAmount = dao.getCurrentAmounts().get("RESTAURANT");
        assertEquals(newAmount, updatedAmount,  delta: 0.01,  message: "Reimbursement amount should be updated in database");
    }
}
```

Sonderfall: Benutzer muss Passwort nach Zurücksetzung ändern

```
@Test
void testMustChangePasswordFlag() {
    User u = new User();
    assertFalse(u.isMustChangePassword(),  message: "Default sollte false sein");
    u.setMustChangePassword(true);
    assertTrue(u.isMustChangePassword());
    u.setMustChangePassword(false);
    assertFalse(u.isMustChangePassword());
}
```

Testplan für Sonderfallbehandlungen (basierd auf Codeanalyse)

ID	Kategorie	Kurzbeschreibung
TC-USER-001	Leere Eingaben	Name oder E-Mail leer beim User hinzufügen
TC-USER-002	Duplikate	Bestehende E-Mail beim User hinzufügen
TC-INV-001	Leere Eingaben	Betrag-Textfeld leer
TC-INV-002	Datei fehlt	Keine Datei beim Hochladen
TC-INV-003	Verbindungsfehler	Supabase offline beim Speichern einer Rechnung
TC-LOGIN-001	Leere Eingaben	Login ohne E-Mail oder Passwort
TC-ADMIN-001	Nicht erlaubt	Zugriff auf Admin ohne Berechtigung (GUI-verhalten)
TC-DB-001	Verbindungsfehler	Keine Datenbankverbindung beim Export oder Import

1. **TC-USER-001 (**Eingabe wird blockiert & Fehlermeldung erscheint**)**

// AddUserController.java (Zeile 37)

```
31      @FXML
32      private void handleAddUser() {
33          String name = nameField.getText().trim();
34          String email = emailField.getText().trim();
35          String role = roleChoiceBox.getValue();
36
37          if (name.isEmpty() || email.isEmpty()) {
38              showAlert( msg: "Name and email cannot be empty.");
39              return;
40          }
41
42          if (UserDAO.emailExists(email)) {
43              showAlert( msg: "Email already exists.");
44              return;
45          }
```

// AddInvoiceController.java (Zeile 52)

```
50      @FXML
51      private void handleSubmit() {
52          if (selectedFile == null || (!restaurantRadio.isSelected() && !supermarketRadio.isSelected())) {
53              showAlert(Alert.AlertType.WARNING,  message: "Please select a file and a category.");
54              return;
55          }
```

2. **TC-INV-002 (Aktion wird nicht ausgeführt & Benutzer erhält Meldung)**

// AdminDashboardController.java (Zeile 173)

```
no usages  zxynep +1
@FXML
private void exportCSV() {
    FileChooser chooser = new FileChooser();
    chooser.setTitle("Save CSV File");
    chooser.setInitialFileName("invoices.csv");
    File file = chooser.showSaveDialog(invoiceTable.getScene().getWindow());
    if (file == null) return;
```

3. **TC-USER-002 (Duplikat wird erkannt & Aktion wird blockiert)**

// AddUserController.java (Zeile 43)

```

    if (UserDAO.emailExists(email)) {
        showAlert( msg: "Email already exists.");
        return;
    }
```

4. **TC-DB-001 (Fehler wird geloggt & Benutzer erhält eine Info)**

// AddInvoiceController.java (Zeile 62)

```
@FXML
private void handleSubmit() {
    if (selectedFile == null || (!restaurantRadio.isSelected() && !supermarketRadio.isSelected())) {
        showAlert(Alert.AlertType.WARNING, message: "Please select a file and a category.");
        return;
    }
    try {
        double invoiceAmount = Double.parseDouble(invoiceAmountField.getText());
        String category = restaurantRadio.isSelected() ? "restaurant" : "supermarket";
        int userId = Session.getCurrentUser().getId();
        double reimbursementAmount = calculateReimbursement(invoiceAmount, category);

        String fileUrl = SupabaseUploadService.uploadFile(selectedFile, userId);
        if (fileUrl != null) {
            boolean success = SupabaseUploadService.saveInvoiceToDatabase(
                    userId, fileUrl, category, invoiceAmount, reimbursementAmount
            );
            if (success) {
                showAlert(Alert.AlertType.INFORMATION, message: "Invoice uploaded and saved successfully.
                closeWindow();
            } else {
                showAlert(Alert.AlertType.ERROR, message: "Failed to save invoice to database.");
            }
        } else {
            showAlert(Alert.AlertType.ERROR, message: "Failed to upload file to Supabase.");
        }
    } catch (NumberFormatException e) {
        showAlert(Alert.AlertType.ERROR, message: "Invalid amount entered. Please enter a valid number.")
    }
}
```