SDLC activity sheets

ACTIVITY SHEET 1 — Requirements Gathering (SRS Development)

Project Title: Password Generator & Validator System

Team Members: 1. T. Someshwari

2. Sukanya Devi Rayapati

3. N. Nandini

4. Vemula Vinay

5. CH .Sahithya

6.Swasthika Battacharya

Task 1: Stakeholder Interview

- End Users
- System Administrator
- Organization IT Team
- Cybersecurity Team
- Developers

Requirement ID	Type	Description
R1	**Functional**	**Generate secure passwords**
R2	**Functional**	**Validate password rules**
R3	**Functional**	**Encrypt passwords**
R4	**Non-Functional**	**Ensure security & speed**

Write your questions below:

1. What type of passwords do users need to generate?
2. Should users be able to customize password length?
3. Should the tool include symbols, numbers, uppercase/lowercase letters?
4. Should the validator follow specific security rules (e.g., minimum length, complexity)?
5. Should the system detect weak/medium/strong passwords?
6. Do we need to store any passwords?
7. Should the UI be simple or feature-rich?
8. Do you expect mobile support?

9. Should users be allowed to copy the password easily?
10. Should the validator show reasons why a Password is week?

Task 2: Extract Requirements

Fill the table:

Requirement ID	Functional / Non-Functional	Requirement Description
R1	F	1.System shall generate strong and random passwords.
R2	F	2.System shall allow the user to select the password length.
R3	F	3.System shall include uppercase letters in the password.
R4	F	4.System shall include lowercase letters in the password.
R4	NF	5.System shall respond to user actions within two seconds.

Task 3: User Stories

Write at least 5.

Format: *As a [user], I want [feature], so that [reason].*

1. As a user, I want to generate strong passwords so that I can secure my accounts.
2. As a user, I want to choose password length so that I can fit different website requirements.
3. As a user, I want to include/exclude symbols or numbers so that I can customize my password.
4. As a user, I want to check password strength so that I know whether it is secure.
5. As a user, I want to see how to improve my password so that I can create a safer one

Deliverable: A mini SRS (1–2 pages)

1.Introduction

The Password Generator & Validator system helps users create strong passwords and validates the strength of existing ones.

2. Functional Requirements

Generate customizable passwords

Validate password strength

Show strength levels

Copy password to clipboard

3. Non-Functional Requirements

Fast response

High usability

No password storage

Cross-platform support

4. System Constraints

Works offline

No user login required

ACTIVITY SHEET 2 — System Design (High-Level & Low-Level)

Project: _Password Generator and Validator_

Task 1: Create a System Architecture Diagram

Draw or attach:

- **Components**

UI Layer

Password Generator Module

Password Validator Module

Strength Analyzer

Clipboard Module

- **Data Flow:**

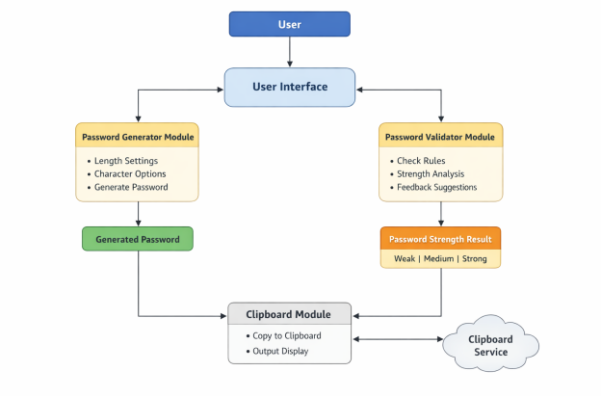
  User → UI → Generator/Validator → Output → UI → User

- **External systems**

User Device (Computer / Mobile)

Operating System

Space for Diagram:



Task 2: Create UI Wireframes

Required screens (tick off as completed):

- Home Screen
- Login / Signup
- Main Feature Screen
- Settings / Profile

Attach wireframes here.

Task 3: Database Design

Create an ERD.

Entity	Attributes
Users	User ID, Username, Email, Encrypted Password
Password Rules	Rule ID, Min Length, Special Char, Uppercase
Logs	Log ID, User ID, Action, Timestamp

Deliverable: Architecture Diagram + UI Wireframes + ERD

ACTIVITY SHEET 3 — Development Phase

Task 1: Break Work Into Tasks

Fill the development backlog:

Task ID	Feature	Description	Assignee	Done
T1	**Password Generator**	**Create strong password logic**	**Sukanya**	
T2	**Validation**	**Validate rules**	**Nandini& Swasthika**	**Done**
T3	**Encryption**	**Encrypt password storage**	**Sahitya& Someshwari**	**Done**

Task ID	Feature	Description	Assignee	Done
T4	UI Integration	Connect UI with backend	Vinay	In Progress

Task 2: Code Walkthrough Notes

After writing code, summarize:

- **Features Implemented:**
- Random password generation
- Rule-based validation
- Password encryption
- **Blockers Faced:**
- Handling special character validation
- **Solutions / Next Steps:**
- Improved regex validation
- UI optimization

Deliverable: Updated backlog + coded feature demo

ACTIVITY SHEET 4 — Testing Phase

Task 1: Create Test Cases

Test Case ID	Description	Steps	Expected Result	Actual Result	Status
TC1	Password Generation	Click generate	Strong password displayed	Same	P
TC2	Weak Password	Enter "123"	Rejected	Rejected	P
TC3	Strong Password	Enter valid password	Accepted	Accepted	P

Task 2: Bug Report

Bug ID	Description	Severity	Steps	Status
B1	Strength meter delay	Low	Enter password	Fixed

Task 3: Test Summary

~ Total test cases: 3

~ Passed: 3

~ Failed: 0

~ Major issues found: None

Deliverable: Test case sheet + bug report

ACTIVITY SHEET 5 — Deployment & Release Notes

Task 1: Deployment Checklist

Tick as completed:

- Code merged
- Database configured
- Environment variables set
- Build successful
- Final testing done
- Version tagged (v1.0, etc.)

Task 2: Release Notes

Write your Version 1.0 release notes.

Release Version:

Features Included:

- Password generator
- Password validation
- Strength indicator
- Encryption support

Known Issues:

- UI enhancement required

Next Update Goals:

- Two-factor authentication
- Password history check

Deliverable: Deployment & Release Document

ACTIVITY SHEET 6 — Maintenance & Reflection

Task 1: Fix & Patch

Record any issues and fixes.

Patch ID	Issue	Root Cause	Fix Implemented	Status
P1	Validation error	Rule mismatch	Updated logic	Done

Task 2: Team Retrospective

What worked well:

- Clear requirements
- Strong security logic

What needs improvement:

- UI responsiveness

Changes for next time:

- Early testing
- Better UI design

Deliverable: Final reflection + updated patch log

Food Ordering App.

ACTIVITY SHEET 1 — Requirements Gathering

Team Members: 1.T. Someshwari

2. Sukanya Devi Rayapati

3. N. Nandini

4. Vemula Vinay

5. CH .Sahithya

6.Swasthika Battacharya

Stakeholder Interview

- End Users
- System Administrator
- Organization IT Team
- Cybersecurity Team
- Developers

2. Extract Requirements

Requirement ID	Type	Description
R1	Functional	System shall generate strong passwords
R2	Functional	System shall validate user passwords
R3	Functional	System shall display password strength
R4	Functional	System shall encrypt passwords
R5	Non-Functional	System should respond quickly
R6	Non-Functional	System must be secure

Requirement ID	Type	Description
R7	Non-Functional	System should be user-friendly

3. User Stories

As a [user], I want [feature], so that [reason].

1. As a user, I want strong passwords so that my account is secure.

2. As a user, I want validation so weak passwords are rejected.

3. As an admin, I want encryption to protect passwords.

4. As a user, I want strength feedback to improve my password.

5. As a system owner, I want rule-based validation for security

Deliverable: Mini SRS

ACTIVITY SHEET 2 — System Design

1. System Architecture Diagram

Sketch or attach the diagram of system components, data flow, and external integrations.

Components

- User Interface
- Password Generator
- Password Validator
- Encryption Module
- Database

Data Flow
User → UI → Generator/Validator → Encryption → Database

2. UI Wireframes

- Home
- Login / Signup
- Menu / Ordering Screen
- Cart / Checkout Screen
- Settings / Profile

Attach sketches or screenshots here.

3. Database Design — ERD

Entity	Attributes
Users	UserID, Name, Email, Phone, Address
Orders	OrderID, UserID, Items, TotalPrice, Status
MenuItems	ItemID, Name, Description, Price, Category

Deliverable: Architecture diagram + wireframes + ERD

ACTIVITY SHEET 3 — Development Phase

Task ID	Feature	Description	Assignee	Status
T1	Password Generator	Generate strong passwords	Sukanya	Done
T2	Validation	Validate password rules	Sahitya	Done
T3	Encryption	Secure password storage	Someshwari	Done

2. Code Walkthrough Notes

Features Implemented:

- **Password generation**
- **Password validation**
- **Encryption**

Blockers Faced:

- **Handling special characters**

Solutions / Next Steps:

- **Improved validation logic**
- **UI improvements**

Deliverable: Updated backlog + coded feature demo

ACTIVITY SHEET 4 — Testing Phase

1. Test Cases

Test Case ID	Description	Steps	Expected Result	Actual Result	Status
TC1	Generate password	Click generate	Strong password shown	Same	P
TC2	Weak password	Enter "123"	Rejected	Rejected	P
TC3	Strong password	Enter valid password	Accepted	Accepted	P

2. Bug Report

Bug ID	Description	Severity	Steps	Status
B1	Strength delay	Low	Enter password	Fixed

3. Test Summary

- **Total test cases: 3**
- **Passed: 3**
- **Failed: 0**
- **Major issues: None**

Deliverable: Test case sheet + bug report

ACTIVITY SHEET 5 — Deployment & Release

1. Deployment Checklist

- Code merged
- Database configured
- Environment variables set
- Build successful
- Final testing done
- Version tagged (v1.0)

2. Release Notes

Release Version: v1.0

Features Included

- **Password generator**
- **Password validation**
- **Encryption**
- **Strength indicator**

Known Issues

- **UI can be enhanced**

Next Update Goals

- **Two-factor authentication**
- **Password history check**

Deliverable: Release document

ACTIVITY SHEET 6 — Maintenance & Reflection

1. Patch Log

Patch ID	Issue	Root Cause	Fix	Status
P1	Validation bug	Rule mismatch	Logic updated	Done

2. Team Retrospective

What worked well

- **Clear requirements**
- **Strong security focus**

What needs improvement

- **UI design**

Changes next time

- **Early testing**
- **Better UI planning**

Deliverable: Final reflection

ACTIVITY SHEET 1 — Requirements Gathering (SRS)

Project Title

Password Generator and Validation System

1. Stakeholders

- End Users (Students / Employees)
- System Administrator
- Organization IT Team
- Cybersecurity Team
- Application Developers
- Database Administrator

Stakeholder Interview Questions (8–10)

1. What problems do users face with weak passwords?
2. What minimum password length should be enforced?
3. Should passwords include special characters?

4. Do users need auto-generated passwords?
5. Should old passwords be restricted from reuse?
6. How should weak passwords be handled?
7. Is real-time password strength feedback required?
8. Should passwords be encrypted before storage?
9. What validation rules are mandatory?
10. Should the system support future security upgrades?

---

2. Extract Requirements

Requirement ID	Type	Description
R1	Functional	System shall generate strong random passwords
R2	Functional	System shall validate user-entered passwords
R3	Functional	System shall display password strength
R4	Functional	System shall encrypt passwords before storage
R5	Non-Functional	System should respond within 2 seconds
R6	Non-Functional	System must ensure data security
R7	Non-Functional	System should be easy to use
R8	Functional	System shall reject weak passwords

---

3. User Stories

1. As a user, I want the system to generate strong passwords, so that my account is secure.
2. As a user, I want password validation, so that weak passwords are rejected.
3. As an admin, I want encryption, so that passwords are protected.
4. As a user, I want strength indicators, so that I know how strong my password is.

5. As a system owner, I want rule-based validation, so that security standards are maintained.

Deliverable: Mini SRS Document

ACTIVITY SHEET 2 — System Design

1. System Architecture (Text Description)

Components:

- User Interface (Web/App)
- Password Generator Module
- Password Validation Module
- Encryption Module
- Database

Data Flow:
User → UI → Generator/Validator → Encryption → Database

---

2. UI Wireframes (Screens)

Home Screen
Login / Signup
Password Generator Screen

Password Validation Screen
Settings / Profile

---

3. Database Design (ERD)

Entity	Attributes
Users	User ID, Username, Email, Encrypted Password
Password Rules	Rule ID, Min Length, Special Char, Uppercase
Logs	Log ID, User ID, Action, Timestamp

Deliverable: Architecture Diagram + Wireframes + ERD

ACTIVITY SHEET 3 — Development Phase

1. Development Backlog

Task ID	Feature	Description	Assignee	Status
T1	Password Generator	Create strong password logic	Sukanya	Done
T2	Validation	Validate rules	Sukanya	Done
T3	Encryption	Encrypt password storage	Sukanya	Done
T4	UI Integration	Connect UI with backend	Sukanya	In Progress

---

2. Code Walkthrough Notes

Features Implemented:

- Random password generation
- Rule-based validation
- Password encryption

Blockers Faced:

- Handling special character validation

Solutions / Next Steps:

- Improved regex validation
- UI optimization

Deliverable: Updated backlog + coded feature demo

ACTIVITY SHEET 4 — Testing Phase

1. Test Cases

Test Case ID	Description	Steps	Expected Result	Actual Result	Status
TC1	Password Generation	Click generate	Strong password displayed	Same	P
TC2	Weak Password	Enter "123"	Rejected	Rejected	P
TC3	Strong Password	Enter valid password	Accepted	Accepted	P

---

2. Bug Report

Bug ID	Description	Severity	Steps	Status
B1	Strength meter delay	Low	Enter password	Fixed

---

3. Test Summary

- Total test cases: **3**
- Passed: **3**
- Failed: **0**
- Major issues found: **None**

Deliverable: Test case sheet + bug report

ACTIVITY SHEET 5 — Deployment & Release Notes

1. Deployment Checklist

Code merged
Database configured
Environment variables set
Build successful
Final testing done
Version tagged (v1.0)

---

2. Release Notes

Release Version: v1.0

Features Included:

- Password generator
- Password validation
- Strength indicator
- Encryption support

Known Issues:

- UI enhancement required

Next Update Goals:

- Two-factor authentication
- Password history check

Deliverable: Deployment & Release Document

ACTIVITY SHEET 6 — Maintenance & Reflection

1. Patch Log

Patch ID	Issue	Root Cause	Fix Implemented	Status
P1	Validation error	Rule mismatch	Updated logic	Done

---

2. Team Retrospective

What worked well:

- Clear requirements
- Strong security logic

What needs improvement:

- UI responsiveness

Changes for next time:

- Early testing

- Better UI design

```
import random
import string

# Function to generate password
def generate_password(length):
    characters = string.ascii_letters + string.digits + string.punctuation
    password = ""
    for i in range(length):
        password += random.choice(characters)
    return password
```

CODE :

```
# Function to validate password
def validate_password(password):
    length_check = len(password) >= 8
    upper_check = any(c.isupper() for c in password)
    lower_check = any(c.islower() for c in password)
    digit_check = any(c.isdigit() for c in password)
    special_check = any(c in string.punctuation for c in password)

    score = length_check + upper_check + lower_check + digit_check + special_check

    if score <= 2:
        return "Weak Password"
    elif score == 3 or score == 4:
        return "Medium Password"
    else:
        return "Strong Password"
```

```
# Main program
print("Password Generator and Validator")

choice = int(input("Enter 1 to Generate Password, 2 to Validate Password: "))

if choice == 1:
    length = int(input("Enter password length: "))
    new_password = generate_password(length)
    print("Generated Password:", new_password)

elif choice == 2:
    user_password = input("Enter password to validate: ")
    result = validate_password(user_password)
    print("Password Strength:", result)

else:
    print("Invalid choice")
```