Term Project

Functional-Spec

CS 151: Object-Oriented Design

Prof: Ahmad Yazdankhah

Team Number	26	**Section**	02
Team Members	• Samriddhi Matharu: Worked on UC-01 Primary Flow and Alternative Flow, as well as the Objective and Scope • Ivan Rivera: (1.2) References, (1.3) Acronyms, and (4) Home Page UI Mockup + define language mockup • Jeremy Greatorex: Problem Statement, Functional/Nonfunctional requirements, Create Student Profile Mockup		
Software Name	StudentSphere		

1 Introduction

1.1 Objective

This document serves as the first draft of the Functional Specification of the Student's Knowledgebase for Faculties software system. The software allows faulty members to record, organize, and retrieve detailed information about their students, including academic status, skills, professional goals, evaluations and service eligibility flags. This system provides a structured and searchable knowledgebase, which helps faculty maintain long term profile of each student, which can better allow them to support future decisions such as recommendation letters and career guidance.

The target audience if this document are the clients (Faculty), software architects, software developers' quality assurance engineers, and project team members who will use this specification as a foundation for design, implementation, and testing.

1.2 References

The references utilized in the creation process of this document include the Project's Problem Statement file, as well as a variety of software programs which assisted with the mockups. Specifically, Moqup's was used to create the Mockup UI Pages and Adobe Suite assisted with the creation of the "StudentSphere" logo!

1.3 Acronyms and Abbreviations

Example: SDLC	Software Development Lifecycle
UC	Use Case

2 Software Overview

2.1 Problem Statement

Throughout their university experience, students take numerous courses with various professors. Their impression, knowledgebase, and experience expressed within these courses is kept only between the student and their single professor. This information is often spread over numerous emails, personal conversations, and assignments out of the eyes of other professors and administration. Evidently, this leaves many professors with little idea of their student's accomplishments or professional experience. Without a software service to log and share this information among common faculty, many professors do not have enough information about their students to write strong recommendation letters or provide applicable career advice.

2.2 Scope

The users of this product are University Faculty Members.

2.3 Functional Requirements

- Data Definition
    - Programming Language List: List defined by faculty of programming languages students can be proficient in
        - Name: Text, Required
    - Student Profile: Describes student, experience, and comments
        - Personal Information:
            - Full Name: Text, Required
            - Year: Select [Freshman, Sophomore, Junior, Senior, Graduate], Required
            - Current Job Status: [Employed, Unemployed]
            - Job Details: Text, Present if Current Job Status is true
        - Experience and Interests: Professional expertise of student
            - Programming Languages Known: [Programming Language List Defined]

        - Databases Known: Select [MySQL, Postgres, MongoDB], Required
        - Preferred Professional Role: Select [Front-End, Back-End, Full-Stack, Data, Other], Required
    - Faculty Evaluation:
        - Comments/Journal Entries: Paragraph Text Entry
- Future Services Flags
    - Whitelist: Boolean, optional (Eligible for recommendations)
    - Blacklist: Boolean, optional (Not Eligible for recommendations)
- Data Entry and Manipulation
    - Edit and delete a student profile
    - Search for students by name or any other parameter in their profile (Case insensitive)
    - Add new comments to an existing student's profile (with date auto stamped)
- Reports
    - List all blacklisted students
    - List all whitelisted students
    - List all students with a specific profile parameter
    - Individual student report containing all details

2.4 Nonfunctional Requirements

- Must be designed with "MVC design pattern"
- Must be desktop application
- Must be single user and no login page
- Must run on Windows, Mac and Linux
- Must be user-friendly, no technical skills required to operate
- All data shall be in flat (OS) files, or in SQLite
- Required fields must be validated before saving
- Name must be decided by development team (us)

3 Use Cases

Use Case Name	UC-01: Define Programming Languages

Goal

Creating a programming language entry so it can be used later in Student Profiles.

Participating Actors

User, System

Glossary

N/A

Primary Flow of Events

Trigger

User clicks on “New Language” button

Steps	Actor's Action	System's Response
1		System redirects user to the **Define Programming Language** page with text field labeled **Name** and **Save** buttons
2	User enters the language’s Name (required)	System captures the input in the text field.
3	User clicks on Save	System validates the **Name** in the field is NOT empty
4		System saves the **Name**.
5		System displays a confirmation message of successful save

Alternative Flow of Events

Alternative Trigger # 1

The user leaves the programming language’s **Name** field empty

Steps	Actor's Action	System's Response
1	User clicks on Save without entering a Name	System prompts user: “Language name is required”

Alternative Trigger # 2

The user enters a programming language **Name** that already exists

Steps	Actor's Action	System's Response
1	User types a duplicate name and clicks **Save**	System prompts user: “Language name already exists”

Alternative Trigger # 3

The user cancels the action

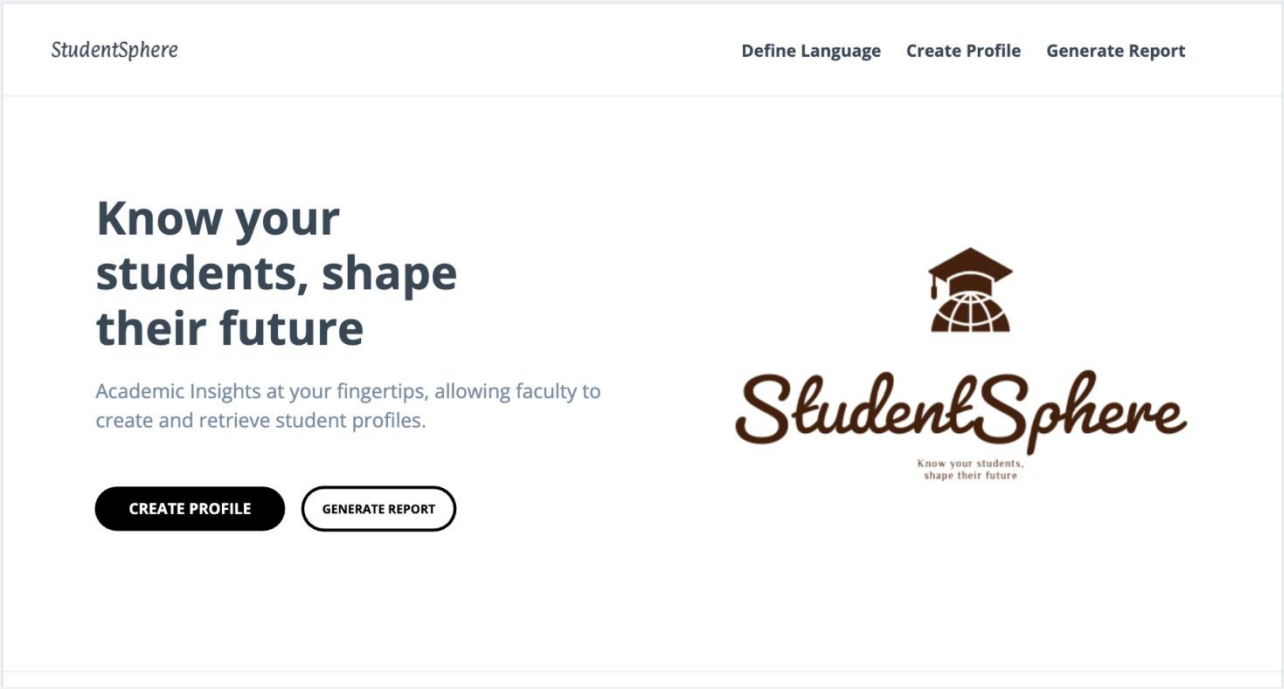
Steps	Actor's Action	System's Response

1	User clicks on **different navigation page on the nav bar header**	System discards the unsaved input and redirects the user back to the chosen page

4 Mockups

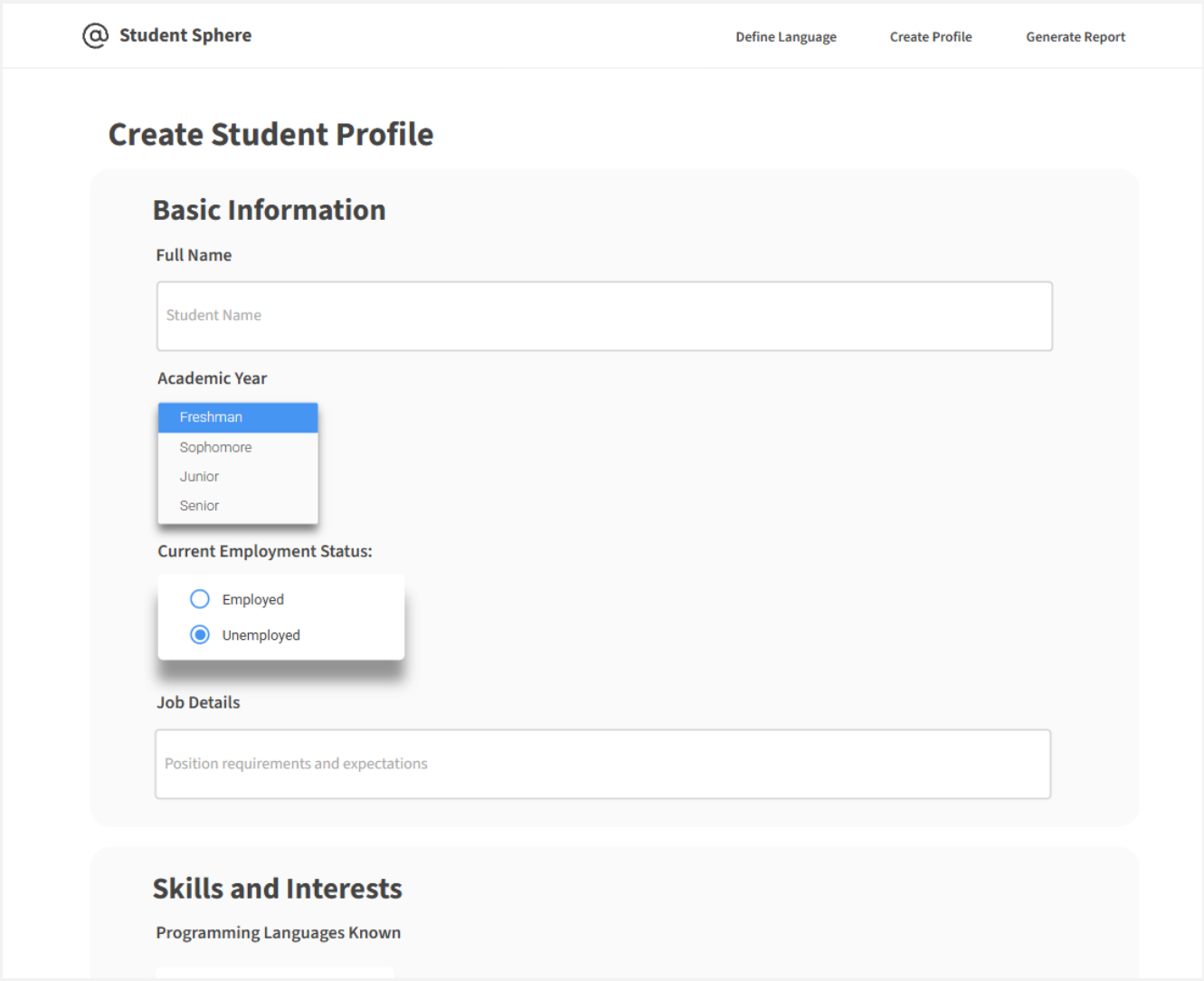
Home Page / Landing Page: Allowing faculty to either

1. Create/Edit Student Profile
2. Define Programming Language
3. Generate Report



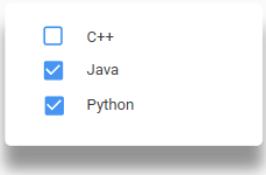
Create Profile Page: Allowing faculty to

1. Define a new student account
2. Navigate to other pages

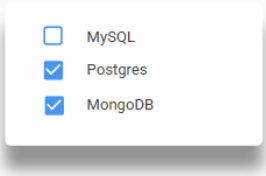

Student Sphere
Define Language
Create Profile
Generate Report
Create Student Profile
Basic Information
Full Name
Student Name
Academic Year
Freshman
Sophomore
Junior
Senior
Current Employment Status:
Employed
Unemployed
Job Details
Position requirements and expectations
Skills and Interests
Programming Languages Known

Skills and Interests

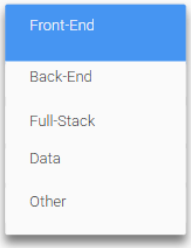
Programming Languages Known



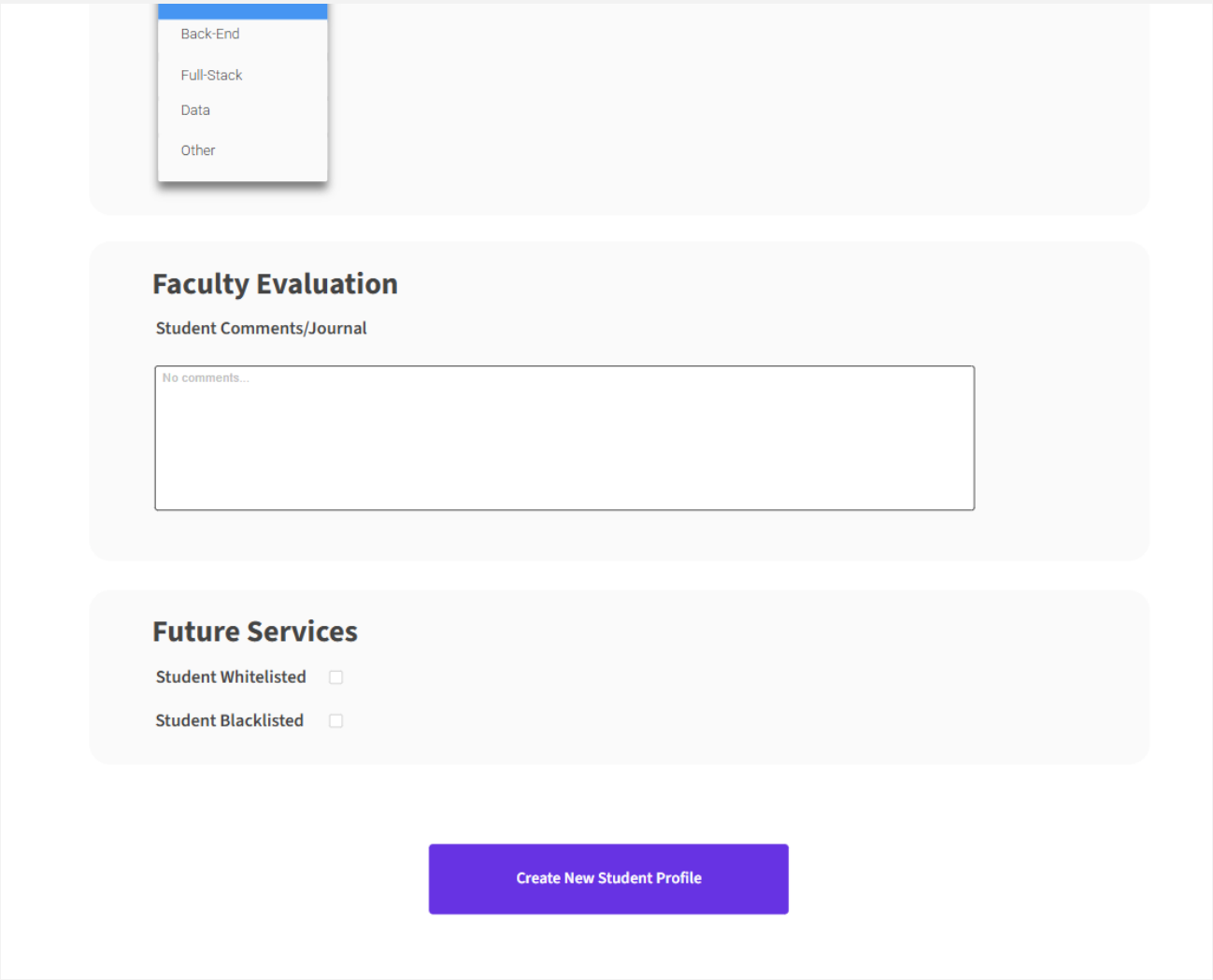
Databases Known



Preferred Professional Role

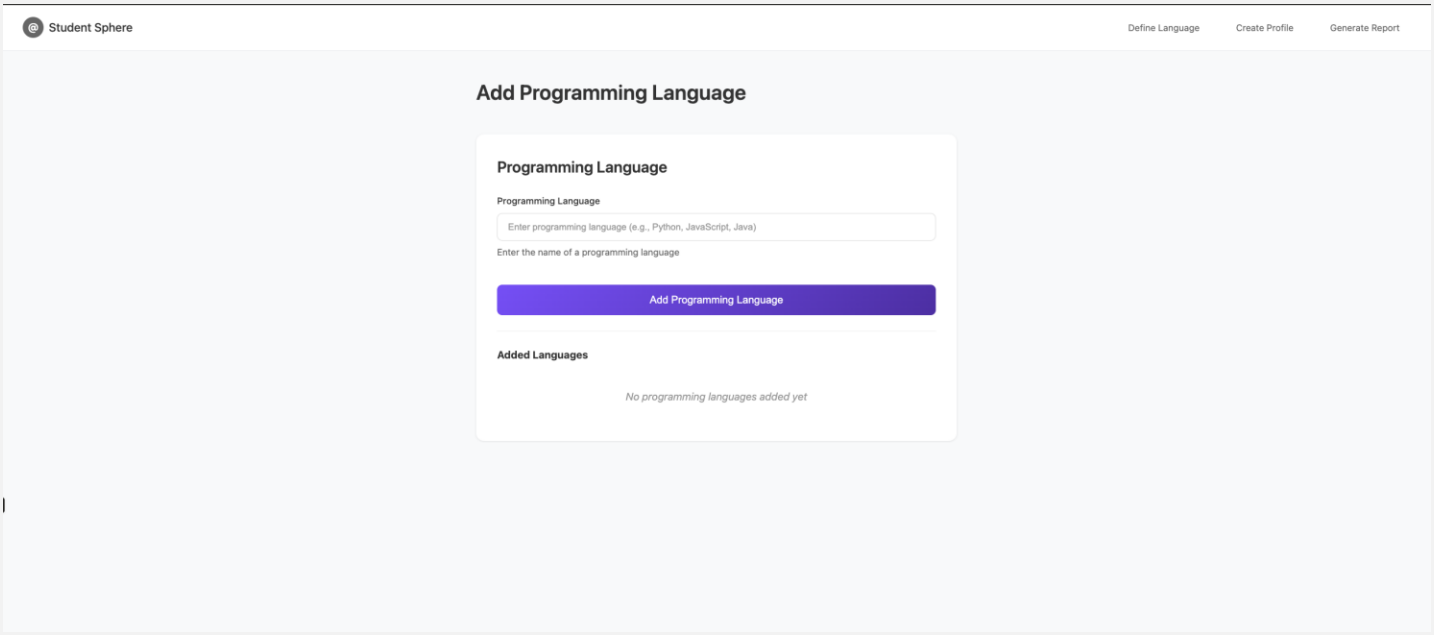


Faculty Evaluation



Define Language Page: Allowing faculty to

1. Define “new” programming language
2. View previously defined programming languages
3. Navigate to other pages


Student Sphere
Define Language
Create Profile
Generate Report
Add Programming Language
Programming Language
Programming Language
Enter programming language (e.g., Python, JavaScript, Java)
Enter the name of a programming language
Add Programming Language
Added Languages
No programming languages added yet