SOFTWARE REQUIREMENT SPECIFICATION FOR STUDENT SATISFACTION SURVEY

NAME	ARAVINTHKUMAR
ROLL NO	7376221CS113
SEAT NO	224
PROJECT ID	24
PROBLEM STATEMENT	STUDENTS SATISFACTION SURVEY

TECHNICAL COMPONENTS:

Front End	• HTML • CSS • JAVASCRIPT
Back End	• Python • Django (Python Web)
Database	• PostgreSQL • MySQL
API	• Open API • SOAP APIs • REST Ful API

PROBLEM STATEMENT:

In educational institutions, understanding student satisfaction is crucial for improving the overall learning experience. However, traditional methods of collecting feedback are often inefficient and lack robust analysis capabilities. Therefore, there is a need for a software solution that streamlines the process of conducting student satisfaction surveys, facilitates comprehensive data analysis, and enables the visualization of survey results through graphical representations.

Purpose:

The purpose of the Student Satisfaction Survey System is to facilitate the systematic collection of feedback from students to improve the overall educational experience. The system aims to provide administrators with actionable insights through graphical representations of survey responses.

Scope:

The scope of the project includes the development of a web-based survey application capable of creating, distributing, collecting, and analysing student satisfaction surveys. The system will generate graphical representations based on survey responses to aid in decision-making processes.

Business Context:

Institutions aim to enhance student satisfaction and improve overall educational quality. By implementing a comprehensive survey system, institutions can gather valuable feedback from students and make data-driven decisions to address areas for improvement.

Consideration:

- The system must comply with relevant privacy regulations to ensure the confidentiality and security of student data.
- User-friendly interfaces and clear instructions should be provided to both administrators and students to facilitate ease of use.
- The system should be scalable to accommodate varying numbers of surveys and survey respondents.

Dependencies:

- Availability of institutional resources for hosting and maintaining the survey system.
- Cooperation from stakeholders (administrators, faculty, students) for successful implementation and adoption.

- Access to survey data analysis tools and techniques for generating meaningful insights.

User Personas:

1. **Administrator:** Responsible for creating, managing, and analysing surveys. Seeks actionable insights to improve institutional processes and student satisfaction.
2. **Student:** The end-user who fills out the survey and provides feedback on their educational experience.

User Stories:

- As an administrator, I want to create customizable surveys with various question types to gather specific feedback from students.
- As a student, I want to receive surveys through accessible channels and provide my feedback conveniently.

FUNCTIONAL REQUIREMENTS:

1. Survey Creation and Management:

- The system shall allow administrators to create and manage surveys.
- Administrators shall be able to define survey questions, including text-based, multiple-choice, and Likert scale questions.
- The system shall support the addition, editing, and deletion of survey questions.
- Surveys may be categorized by academic term, course, department, or other relevant criteria.

2. Survey Distribution:

- The system shall provide mechanisms for distributing surveys to students electronically.
- Surveys may be distributed through a web-based interface.
- Surveys should be accessible to students for a defined period, with reminders sent to non-respondents.

3. Survey Deployment:

- The software shall enable administrators to deploy surveys to specific groups of students or to the entire student body.
- Surveys should be accessible through a secure login system to ensure data integrity and confidentiality.
- Administrators shall have the ability to set deadlines for survey completion.

4. Survey Submission:

- Students should be able to access the survey through a user-friendly interface.
- The system shall validate survey responses to ensure completeness and accuracy.
- Students must have the option to save their progress and return to complete the survey later.

5. Data Analysis:

- The software shall compile survey responses into a structured database for analysis.
- Administrators should be able to view survey results in real-time.
- The system shall provide statistical analysis tools to identify trends and patterns in the data.

6. Graphical Representation:

- The software shall generate graphical representations (e.g., bar charts, pie charts) based on survey responses.
- Administrators should be able to customize the appearance of graphs (e.g., colours, labels).
- Graphs should be dynamically updated as new survey responses are submitted.

NON-FUNCTIONAL REQUIREMENTS:

1. Usability:

- The software interface shall be intuitive and easy to navigate for both administrators and students.
- Response forms should be mobile-friendly to accommodate users accessing the survey from various devices.

2. Performance:

- The system shall be capable of handling a large volume of survey responses simultaneously.
- Response times for survey creation, distribution, and data analysis should be minimized to provide a responsive user experience.

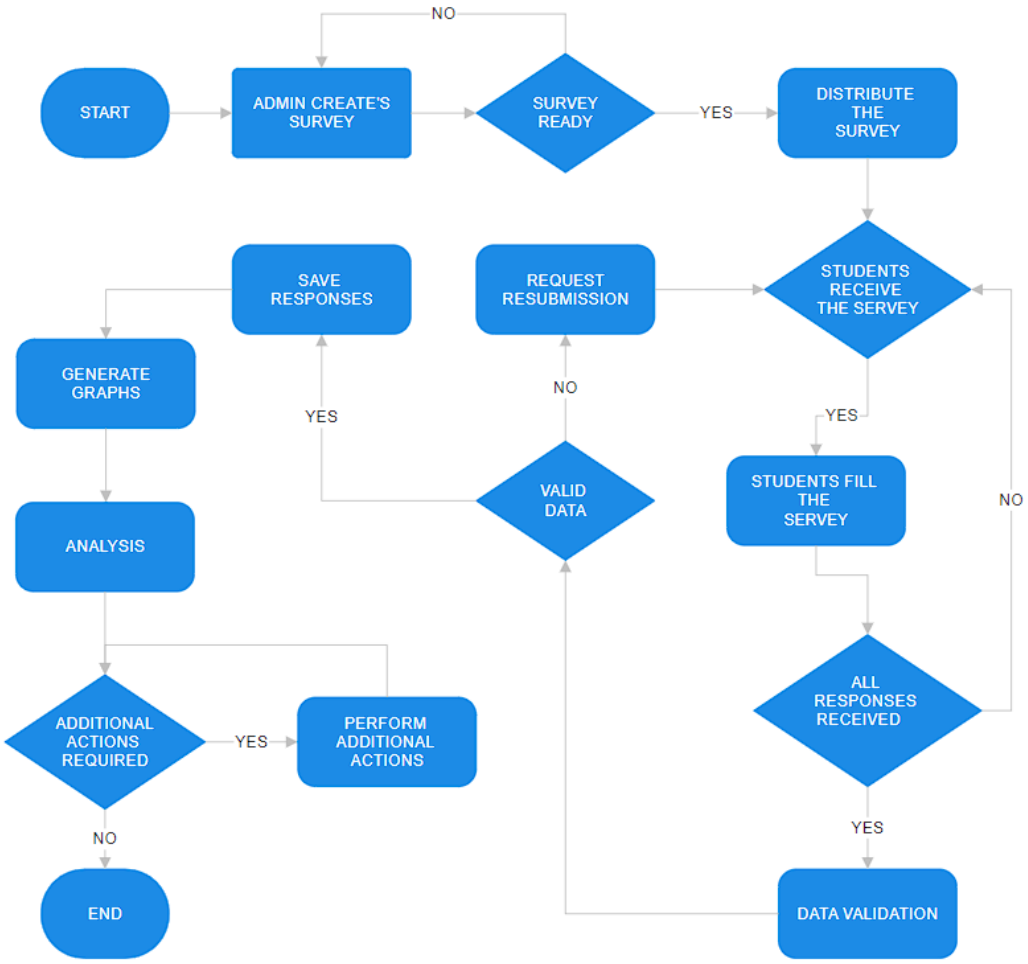
3. Security:

- The system shall implement robust authentication and authorization mechanisms to ensure only authorized users can access survey data.
- Survey data must be encrypted both in transit and at rest to protect confidentiality.
- Role-based access control shall be implemented to restrict administrative privileges based on user roles.

4. Reliability:

- The software shall have built-in mechanisms for data backup and recovery to prevent loss of survey responses.
- The system should be highly available, with minimal downtime for maintenance or upgrades.

FLOWCHART:



PROJECT WORK-FLOW:

1. Project Planning and Requirement Gathering:

- Define project goals, scope, and objectives.
- Gather detailed requirements from stakeholders, including features like survey creation, distribution, analysis, and visualization.
- Identify technical stack: Django, PostgreSQL for the backend; HTML, CSS, JavaScript for the frontend.
- Plan the project timeline, milestones, and resource allocation.

2. Design Phase:

Backend Design:

- Design the database schema with tables for surveys, questions, responses, and user roles.
- Outline the API structure and endpoints for CRUD operations on surveys.

Frontend Design:

- Create wireframes and mockups for user interfaces (admin dashboard, student survey pages).
- Plan the user flow and interaction design, focusing on ease of use for both administrators and students.
- Decide on the visual style, colour schemes, and responsive design elements.

3. Development Phase:

Backend Development:

- Set up the Django project and configure PostgreSQL.
- Implement models, views, and serializers for survey-related data.
- Develop authentication and authorization features with role-based access.

Frontend Development:

- Build the HTML, CSS, and JavaScript components for the user interface.
- Integrate frontend with the Django backend using AJAX or Fetch API for dynamic content loading.
- Ensure the frontend is responsive and mobile-friendly.

4. Testing Phase:

Backend Testing:

- Write and execute unit tests for models, views, and APIs.
- Perform integration testing to ensure smooth interaction between different modules.

Frontend Testing:

- Conduct cross-browser and cross-device testing to ensure compatibility.

- Perform user acceptance testing (UAT) with stakeholders to validate functionality against requirements.

Security Testing:

- Test for vulnerabilities like XSS, CSRF, SQL injection, and ensure data encryption and secure authentication.

5. Deployment Phase:

Preparation:

- Set up a production environment using Docker, Nginx, and Gunicorn.
- Prepare deployment scripts and configuration files.

Deployment:

- Deploy the application to a cloud service like AWS or Heroku.
- Migrate the database to PostgreSQL and ensure all data is correctly configured.
- Monitor the initial deployment for performance and security issues.

6. Maintenance and Update Phase:

Ongoing Support:

- Monitor the system for bugs, performance issues, and security threats.
- Regularly update the application with new features, improvements, and security patches.
- Gather feedback from users to continuously improve the system.
- Perform regular backups, update dependencies, and ensure the system remains scalable and reliable.

CONCLUSION:

The Students Satisfaction Survey software outlined in this document aims to provide a comprehensive solution for collecting, analysing, and presenting feedback from students. By adhering to the outlined requirements, the system will facilitate the efficient creation, distribution, and analysis of surveys, ultimately enhancing the quality of education and student engagement.