Software Design Specifications

for

ProfInsights v1.0

Prepared by: Team ProfInsights, Mahindra University

Document Information

Title: Software Design Specifications for ProfInsights
Project Manager: Attem Niharika SE22UCSE038
Niharika Mallavarapu – SE22UCSE182
Sanjana Arroju – SE22UCSE309
Siri Vennela – SE22UCSE212
Divya Reddy – SE22UCSE083
Ananya Gandla – SE22UCSE025

Document Version No: 1.0
Document Version Date: April 7, 2025
Prepared By: Group 8
Preparation Date: April 7, 2025
Version History
Ver. No: 1.0 Ver. Date: April 7, 2025
Revised By: Group 8
Description: Initial Complete Draft
Filename: ProfInsights_SDS_v1.0.docx

1 INTRODUCTION

1.1 PURPOSE

This document provides the detailed software design specifications for the ProfInsights application. It guides the development team in building the system architecture and modules that meet the software requirements. This document is intended for developers, testers, project managers, and academic supervisors.

1.2 SCOPE

ProfInsights is a web platform that enables students to anonymously rate and review professors based on various teaching and course engagement criteria. This document focuses on frontend design aspects, component architecture, data models, exception handling, and configuration parameters.

1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

- UI: User Interface
- API: Application Programming Interface
- CRUD: Create, Read, Update, Delete

- SRS: Software Requirements Specification
- SDS: Software Design Specification
- DB: Database
- SPA: Single Page Application

1.4 REFERENCES

- Software Requirements Specification for ProfInsights
- React.js Documentation
- Tailwind CSS Documentation
- Node.js and Express Documentation
- MongoDB Documentation

2 USE CASE VIEW

2.1 USE CASE Use Case 1: Submit Review

- Description: Student submits an anonymous review for a professor
- Steps: Navigate to professor profile > Click "Write a Review" > Fill star ratings and review text > Submit

Use Case 2: Search Professor

- Description: Student searches for a professor by name, course, or university
- Steps: Enter search term on homepage > View filtered list > Select professor

3 DESIGN OVERVIEW

3.1 DESIGN GOALS AND CONSTRAINTS

- Use a responsive and modern frontend framework (React.js)
- Ensure accessibility and mobile-first UI
- Implement real-time UI updates using API integration
- Adhere to RESTful API standards
- Style using Tailwind CSS for modular, utility-based design

3.2 DESIGN ASSUMPTIONS

- Users will not need to log in
- All reviews are anonymous

- Moderation is handled by an AI backend
- The system assumes stable internet connectivity

3.3 SIGNIFICANT DESIGN PACKAGES

- Components: Navbar, ReviewCard, StarRating, ProfessorProfile
- Pages: Home, ProfessorList, SubmitReview
- Services: API Service for GET/POST review data

3.4 DEPENDENT EXTERNAL INTERFACES

External Module	**Interface Name**	**Description**
MongoDB	ReviewCollection	Stores and retrieves review data
Express API	/api/review	Accepts POSTed review data
Express API	/api/professors	Delivers list and details of professors

3.5 IMPLEMENTED APPLICATION EXTERNAL INTERFACES

Interface Name	Module Implementing	Description
/api/review	ReviewService	Handles review submission to DB
/api/professors	ProfessorService	Retrieves professor data for UI rendering

4 LOGICAL VIEW

4.1 DESIGN MODEL

- App.js: Root component handling routing and layout
- Home.js: Displays intro and top-rated professors
- ProfessorList.js: Renders filtered professor cards
- ProfessorProfile.js: Shows ratings and review submission form
- SubmitReview.js: Handles review input and POST to backend

4.2 USE CASE REALIZATION

- Sequence Diagram for Submit Review: User > UI > API > DB > Confirmation to User
- Sequence Diagram for Search: User > UI > API > Filtered Response > Render

5 DATA VIEW

5.1 DOMAIN MODEL Entities:

- Professor: { id, name, department, averageRating, reviewIds[] }
- Review: { id, professorId, rating[], comment, timestamp }

5.2 DATA MODEL (PERSISTENT DATA VIEW) MongoDB collections:

- professors
- reviews

5.2.1 DATA DICTIONARY

Field	Type	Description
id	String	Unique identifier
name	String	Professor name
rating	Number[]	Array of 1–5 star ratings
comment	String	User's review
timestamp	Date	Submission date
professorId	String	Foreign key to Professor entity

6 EXCEPTION HANDLING

- Handle network errors (API call fails)
- Validate review form before submission
- Handle empty responses for search
- Show error messages for server failures

7 CONFIGURABLE PARAMETERS

Configuration Parameter Name	Definition and Usage	Dynamic?
API_BASE_URL	Backend endpoint URL	Yes
MAX_REVIEW_LENGTH	Limit review characters	Yes
RATING_CRITERIA_COUNT	Number of star categories	Yes

8 QUALITY OF SERVICE

8.1 AVAILABILITY

- Target: 99.9% uptime

- Uses cloud hosting with auto-scaling

8.2 SECURITY AND AUTHORIZATION

- No login required (anonymous usage)
- Server performs moderation using AI models
- Review data sanitized before display

8.3 LOAD AND PERFORMANCE IMPLICATIONS

- Optimized for 1M concurrent users
- Real-time updates using async API calls
- Data caching handled via backend (Redis or equivalent)

8.4 MONITORING AND CONTROL

- Logs frontend errors using monitoring services like Sentry
- Status UI components reflect system health (e.g., loading spinners, error alerts)