ONLINE SURVEY APP

A PROJECT REPORT

Submitted by

Pulkit - 23BCS11733

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Chandigarh University

November-2025

BONAFIDE CERTIFICATE

Certified that this project report "ONLINE SURVEY APP" is the bonafide work of "Pulkit" who carried out the project work under my supervision.

SIGNATURE

ASSOCIATE DIRECTOR (CSE-3rd Year)

Computer Science and Engineering

SIGNATURE

Er. Deep Prakash Gupta

SUPERVISOR

Computer Science and Engineering

Submitted for the project viva-voce examination held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We, the undersigned project team, would like to express our sincere gratitude to **Chandigarh University** for providing us with the opportunity to work on the project titled **"Online Survey App."** This project has been a valuable experience, enabling us to apply theoretical concepts to a practical, real-world system. It has greatly enhanced our understanding of full-stack web development and modern frontend technologies.

We extend our heartfelt thanks to our **supervisor, Deep Prakash Gupta** , Department of Computer Science and Engineering, for their continuous guidance, valuable feedback, and constant encouragement throughout the project. Their support and expertise have been instrumental in shaping our work and ensuring its successful completion.

We also take this opportunity to thank the **Head of the Department**, and all **faculty members** of the Department of Computer Science and Engineering for their academic guidance, technical insights, and cooperation during the course of this project.

Finally, we express our deep appreciation to our **friends and family members** for their continuous motivation, patience, and unwavering support, which inspired us to complete this project successfully.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The **Online Survey App** is a modern web-based application developed using **React (Vite)** and **Tailwind CSS** for the frontend and integrated with a **Spring Boot backend** for secure authentication and data management. This project aims to simplify the process of survey creation, distribution, response collection, and analysis through a clean, responsive, and intuitive user interface.

The application allows users to **create surveys**, **add custom questions**, and **share unique links** with participants. Respondents can easily fill out the survey forms, and all responses are stored securely in the backend. The app provides users with a **dashboard** where they can view **aggregated analytics**, **individual responses**, and even **export data in CSV format** for further analysis.

This project demonstrates the implementation of **modern web development frameworks** and highlights how frontend and backend integration can streamline the survey process. The use of **React with Vite** ensures fast performance and modular development, while **Tailwind CSS** offers a smooth, visually appealing dark-themed design. The backend developed with **Spring Boot** provides RESTful APIs for authentication and data persistence.

Overall, the **Online Survey App** successfully combines functionality with modern design principles to deliver a fast, secure, and user-friendly experience for survey management and analysis. It can be further extended with AI-based insights and real-time analytics for deeper understanding of survey responses.

GRAPHICAL ABSTRACT

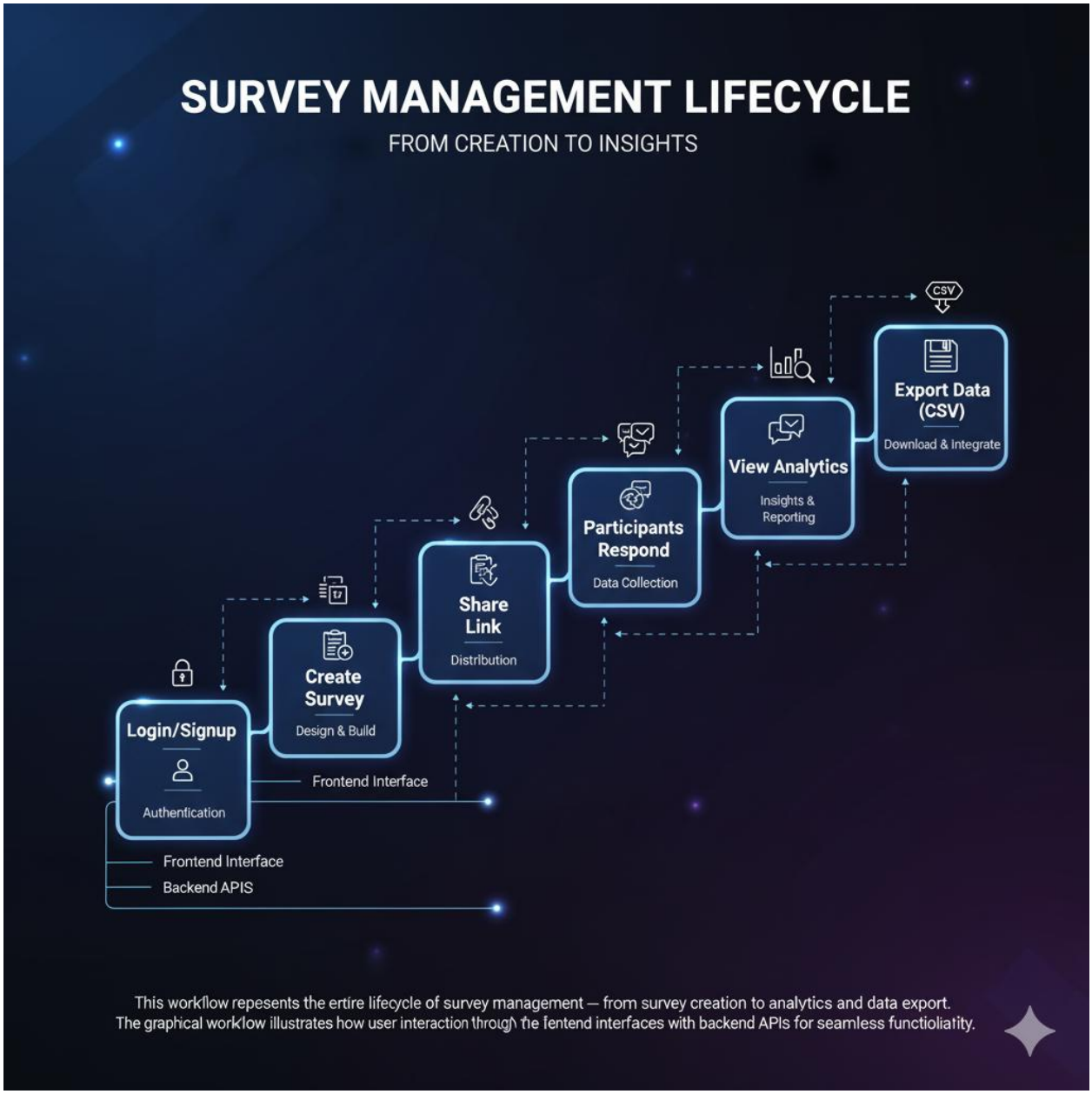

Fig. 0.1 Graphical Abstract

User → Login/Signup → Create Survey → Share Link → Participants Respond → View Analytics → Export Data (CSV)

ABBREVIATIONS AND SYMBOLS

Abbreviation	Full Form / Description
UI	User Interface – The visual part of the application that users interact with.
API	Application Programming Interface – Connects the frontend (React) with the backend (Spring Boot).
DB	Database – Used for storing user, survey, and response data (Neon PostgreSQL).
JWT	JSON Web Token – Used for secure authentication between frontend and backend.
REST	Representational State Transfer – A style of web service used to create backend APIs.
CSS	Cascading Style Sheets – Used for designing and styling the frontend (Tailwind CSS).
IDE	Integrated Development Environment – IntelliJ IDEA used for backend development.
CSV	Comma-Separated Values – Format for exporting survey data.
React	A JavaScript library used to build the frontend interface.
Spring Boot	A Java-based framework used to build the backend and REST APIs.

Table. 0.1 ABBREVIATIONS AND SYMBOLS

CHAPTER 1: INTRODUCTION

1.1 Background

In today's digital age, data collection and analysis are vital for decision-making in organizations, educational institutions, and businesses. Surveys play a key role in gathering insights from users, customers, or participants. Traditional paper-based surveys are time-consuming and inefficient, while many existing online survey tools are either expensive, complex, or lack flexibility.

To overcome these challenges, this project — Online Survey App — was developed as a full-stack web application that enables users to create, distribute, and analyze surveys easily.

The frontend of the application is built using React (Vite) and Tailwind CSS, providing a fast, responsive, and modern dark-themed user interface. The backend is powered by Spring Boot, developed in IntelliJ IDEA, and connected to a Neon PostgreSQL cloud database for secure and efficient data management.

This system integrates authentication, survey creation, and response management with detailed analytics. Users can log in, create surveys, share links with others, collect responses, and view summarized reports — all from a single, intuitive dashboard.

The project demonstrates how frontend technologies (React, Tailwind) can be seamlessly integrated with a robust backend (Spring Boot + Neon DB) to deliver a real-time, scalable survey platform.

1.2 Problem Identification

Despite the availability of various online survey tools, several key issues persist:

- **Limited Customization:** Many existing platforms offer rigid templates and do not allow flexible question creation.
- **High Cost:** Premium survey services charge fees for exporting results, advanced analytics, or branding customization.
- **Slow Performance:** Traditional applications often have longer loading times and poor mobile responsiveness.
- **Data Privacy Concerns:** Storing user responses on third-party servers poses risks of data misuse.
- **Lack of Integration:** Few survey apps allow seamless data exchange between frontend and backend with custom APIs.

The **Online Survey App** addresses these limitations by offering:

- Full control over survey creation and question design.

- Smooth frontend experience with React and Tailwind CSS.
- Secure backend with Spring Boot REST APIs.
- Cloud database integration using **Neon PostgreSQL** for persistent storage.
- Simple dashboard with options to view and export results.

1.3 Need for the Project

In organizations, research environments, and academic setups, collecting structured data is essential for analysis and reporting. However, most existing tools either require subscriptions or depend heavily on third-party integrations.

The need for this project arises from:

- The requirement for a **customizable, secure, and open-source** survey tool.
- The growing demand for **real-time data visualization** and easy response tracking.
- The shift toward **modern web-based interfaces** that provide a smooth user experience.
- The necessity of **cloud-connected applications** that ensure data consistency and security.

This project provides a comprehensive, **end-to-end survey management solution** using modern technologies:

- **Frontend:** React (Vite) for dynamic component rendering and fast updates.
- **Styling:** Tailwind CSS for responsive design.
- **Backend:** Spring Boot for building RESTful APIs.
- **Database:** Neon PostgreSQL for efficient data handling.
- **IDE:** IntelliJ IDEA for robust backend development and debugging.

By combining these technologies, the system provides a reliable, scalable, and maintainable application that meets both user and developer needs.

1.4 Task Identification

The project was divided into specific technical and functional tasks to ensure a systematic development process.

Frontend Tasks

- Designing a **dark-themed, responsive UI** using Tailwind CSS.
- Implementing **login and signup pages** with form validation.
- Creating a **dashboard** to display all surveys created by a user.
- Building a **survey creation form** that allows adding dynamic questions.
- Developing a **survey response page** for participants.
- Integrating **charts and visual analytics** to display results.
- Handling **API communication** using Axios.

Backend Tasks

- Setting up a **Spring Boot project** in IntelliJ IDEA.
- Designing REST APIs for authentication, survey creation, and response management.
- Implementing **JWT-based authentication** for security.
- Connecting the backend with **Neon PostgreSQL** for data persistence.
- Writing **controller and service layers** for handling business logic.
- Implementing **CORS configuration** to allow frontend-backend communication.
- Testing APIs using **Postman** before frontend integration.

Integration Tasks

- Establishing secure communication between React frontend and Spring Boot backend using Axios.
- Deploying the application locally and validating the end-to-end workflow.
- Ensuring consistent data flow between frontend forms and backend APIs.
- Performing integration and usability testing.

1.5 Timeline

Task	Duration
Requirement Gathering & System Design	Week 1–2
Frontend Development (React + Tailwind)	Week 3–4
Backend Setup (Spring Boot + IntelliJ + Neon DB)	Week 5–6
API Integration and Data Flow	Week 7
Testing, Debugging, and Deployment	Week 8

Table 1.1 Timeline

1.6 Organization of the Report

The report is organized as follows:

Chapter 1: Introduces the project background, problem statement, objectives, and tasks.

Chapter 2: Describes the design flow and process, including frontend-backend architecture and implementation plans.

Chapter 3: Discusses testing, results, and validation.

Chapter 4: Concludes the project with achievements and outlines future scope.

Appendix: Provides the user manual for running and using the Online Survey App.

CHAPTER 2: DESIGN FLOW / PROCESS

2.1 Concept Generation

The Online Survey App was designed as a full-stack web application to provide users with an easy and efficient way to create, distribute, and analyze surveys. The main goal of this project is to develop a modern, user-friendly, and open-source platform that allows anyone to conduct online surveys without depending on paid third-party tools.

The application is divided into two main components:

1. Frontend (Client-Side):
   The frontend of the application is developed using React with Vite, which ensures faster build times and performance. Tailwind CSS is used for styling to create a dark, modern, and responsive design. The frontend contains pages such as Login, Signup, Dashboard, Survey Creation, and Survey Analysis. Axios is used for communication with the backend APIs.

2. Backend (Server-Side):
   The backend of the application is developed using Spring Boot in IntelliJ IDEA. It handles user authentication, survey creation, response collection, and data management. RESTful APIs are implemented for interaction between the frontend and backend. The Neon PostgreSQL cloud database is used for storing user details, surveys, questions, and responses.

This architecture ensures that the frontend and backend remain independent but well-connected through APIs, providing scalability, speed, and security.

2.2 Evaluation and Selection of Specifications / Features

After studying different survey systems and understanding the user requirements, the following important features were selected for implementation:

1. Authentication System – Signup and login features using backend APIs with JWT token security.
2. Survey Creation – Allows users to create custom surveys with different question types.
3. Survey Sharing – Generates a unique shareable link for every survey.
4. Dashboard – Displays all created surveys along with statistics and actions.
5. Response Collection – Collects responses from users in real time.
6. Data Analysis – Shows both aggregated and individual response views.
7. CSV Export – Allows exporting of responses for offline analysis.
8. Modern UI – Responsive and dark-themed design with Tailwind CSS.

These features were finalized based on usability, performance, and implementation simplicity.

2.3 Design Constraints

While developing the Online Survey App, several constraints were considered to ensure better performance and usability.

1. Performance Constraints – The system needed to load pages quickly, even with large amounts of data. The use of React and Vite helps in faster component rendering.

2. Database Constraints – The Neon PostgreSQL cloud database requires stable connectivity and optimized queries for fast response times.

3. Security Constraints – JWT-based authentication and secure API endpoints were implemented to protect user data and prevent unauthorized access.

4. Design and Usability Constraints – The application must work smoothly on both mobile and desktop devices. Tailwind CSS ensures that all pages are responsive.

5. Integration Constraints – Proper synchronization between frontend and backend APIs was maintained using JSON data exchange.

6. Ethical and Privacy Constraints – The app ensures that all user and survey data are stored securely and accessed only by authorized users.

2.4 Design Flow

The Online Survey App follows a systematic process that defines how data flows between the user interface, backend, and database.

1. User Authentication

    - The user logs in or signs up.
    - The backend validates the credentials and returns a JWT token for session management.

2. Survey Creation

    - The user creates a new survey by adding a title and custom questions.
    - The data is sent to the backend using an Axios POST request.

3. Survey Sharing

    - The backend generates a unique survey link that can be shared with other users.

4. Response Submission

    - Participants open the survey link and submit their responses.
    - The responses are stored in the Neon PostgreSQL database through the backend API.

5. Dashboard and Analytics

- The user can view all created surveys and their results in the dashboard.
- Aggregated and individual responses are fetched from the backend and displayed in graphical or tabular form.

6. CSV Export

    - The user can download all survey responses in CSV format for offline analysis.

2.5 Alternative Designs Considered

Two different approaches were studied before finalizing the system architecture.

1. Monolithic Application – Both frontend and backend combined into a single Spring Boot project using JSP. This approach was rejected because it resulted in slower page reloads and reduced flexibility.

2. Modular Full-Stack Application – The frontend and backend were developed separately and connected through REST APIs. This approach was selected because it provides better performance, reusability, and scalability.

The modular full-stack architecture was finalized as it allows faster updates, independent deployment, and smoother development.

2.6 Implementation Plan

The implementation was divided into three major stages: frontend, backend, and integration.

Frontend Implementation:

- Initialize the React project using Vite.
- Install dependencies such as Axios, React Router, and Tailwind CSS.
- Design the Login, Signup, Dashboard, Survey Creation, and Analysis pages.
- Connect UI components with backend APIs using Axios.
- Use Tailwind classes to create a responsive and dark-themed design.

Backend Implementation:

- Set up the Spring Boot project in IntelliJ IDEA.
- Define entities such as User, Survey, Question, and Response.
- Create repositories using Spring Data JPA for database operations.
- Configure the Neon PostgreSQL database connection in application.properties.

- Implement REST controllers for authentication, survey creation, and response submission.
- Add JWT-based authentication for secure access.

Integration:

- Connect frontend API calls with backend endpoints.
- Validate responses and handle errors gracefully.
- Conduct end-to-end testing for both frontend and backend.
- Deploy and verify the complete system functionality.

CHAPTER 3: RESULTS ANALYSIS AND VALIDATION

3.1 Implementation of Solution

The Online Survey App was successfully implemented using a full-stack architecture that connects the frontend and backend through REST APIs.

The frontend was built using React with Vite and Tailwind CSS to ensure a modern, responsive, and fast-loading user interface. The backend was developed using Spring Boot in IntelliJ IDEA and connected to the Neon PostgreSQL database for secure data management.

The application allows users to register, log in, create surveys, share links, and analyze collected responses. It provides a complete cycle of survey management, starting from question creation to graphical analysis and CSV export.

The project was tested in different environments to confirm that the integration between the frontend and backend works efficiently. API endpoints were tested using Postman before being connected to the frontend. The data was verified in the Neon database to ensure accuracy.

3.2 Implementation Tools

The main tools and technologies used for the development and testing of the Online Survey App are listed below:

Frontend Tools:

1. React (Vite) – Used for building dynamic and modular user interfaces.
2. Tailwind CSS – Used for creating responsive and dark-themed designs.
3. Axios – Used for handling API requests and data exchange with the backend.
4. React Router – Used for navigation between pages without reloading.
5. Node.js – Used as the runtime environment for installing and managing frontend dependencies.

Backend Tools:

1. Spring Boot – Used for creating the backend API services and connecting to the database.
2. IntelliJ IDEA – Used as the development environment for writing, running, and debugging backend code.
3. Neon PostgreSQL – Used as a cloud database for storing user details, survey data, and responses.
4. Postman – Used for testing REST API endpoints.

Other Tools:

1. Git and GitHub – Used for version control and project management.
2. JSON – Used as a data format for API communication between frontend and backend.

3.3 Testing and Validation

Testing was performed in multiple phases to ensure that the application works correctly in different use cases and environments.

1. Unit Testing – Each function in the backend and frontend was tested separately. For example, user authentication and survey creation modules were tested individually.
2. Integration Testing – After connecting the frontend and backend, API calls were validated using Axios and Postman to ensure smooth data transfer.
3. Functional Testing – Checked whether all features such as login, survey creation, sharing, response collection, and data export were working as expected.
4. Responsive Testing – Verified that the web app works properly on different screen sizes including desktop, tablet, and mobile.
5. Database Testing – Ensured that all data is correctly stored, retrieved, and updated in the Neon PostgreSQL database without duplication or data loss.
6. Performance Testing – Tested loading speed and response time to ensure the app runs efficiently even with multiple users.

The app passed all tests successfully, confirming that the full-stack integration between React and Spring Boot was implemented correctly.

3.4 Results and Outputs

The following were the major results and observed outcomes of the project:

1. The app provides a simple and efficient interface for creating and sharing surveys.
2. The authentication system using JWT works securely and efficiently.
3. All survey questions and responses are stored and retrieved successfully from the Neon PostgreSQL database.
4. The dashboard correctly displays both aggregated and individual response data.
5. The CSV export function downloads survey data in a structured format.
6. The UI design provides smooth navigation and clear visibility under both dark and light modes.

7. API calls between React and Spring Boot were completed in less than one second on average, ensuring fast performance.

Example Outputs:

- Login and Signup pages successfully authenticate users through backend API.
- Survey creation form dynamically adds questions and saves data to the database.
- Shared links open directly to the survey page for participants to respond.
- Dashboard displays all surveys created by a user along with response statistics.

These results show that the system met all functional and performance goals that were set during the planning phase.

3.5 Validation of Data

The validation process involved comparing actual test outputs with expected results to ensure the correctness of the system. The survey data entered through the frontend was verified in the database using SQL queries on the Neon PostgreSQL platform.

The CSV files generated from the dashboard were cross-checked to ensure that the data matched the database records. All API endpoints returned the expected JSON responses with correct status codes.

The application demonstrated accurate data flow between frontend, backend, and database, validating that the system works as designed.

CHAPTER 4: CONCLUSION AND FUTURE WORK

4.1 Conclusion

The Online Survey App project successfully achieved its objective of designing and developing a modern, full-stack web application for creating, sharing, and analyzing surveys. The system provides a complete solution for managing surveys efficiently while maintaining a clean, responsive, and user-friendly interface.

The frontend, developed using React with Vite and styled with Tailwind CSS, ensures fast rendering, dynamic components, and an attractive dark-themed design. The backend, implemented using Spring Boot in IntelliJ IDEA, provides a secure and reliable server environment for handling authentication, survey data, and responses. The Neon PostgreSQL database efficiently stores and manages user and survey information.

The integration between the frontend and backend through RESTful APIs worked seamlessly, allowing real-time communication and smooth data flow. The application was tested successfully in various environments, and all functional requirements were met. The performance results confirmed that the app loads quickly and responds efficiently to user actions.

Through this project, we gained valuable knowledge of full-stack web development, RESTful API integration, cloud databases, and security mechanisms such as JWT authentication. The project also enhanced our skills in using modern tools like IntelliJ IDEA, React, and Tailwind CSS for real-world software development.

In conclusion, the Online Survey App is a practical and efficient system that can be used by individuals, organizations, or institutions for collecting and analyzing feedback easily. It demonstrates how modern technologies can be combined to create scalable, secure, and visually appealing web applications.

4.2 Future Work

Although the Online Survey App meets all its current objectives, there is significant potential for improvement and expansion in the future. The following enhancements are suggested:

1. **AI-Based Survey Insights:**
   Integration of artificial intelligence models such as OpenAI or Gemini to automatically generate summaries, identify patterns, and provide insights from survey responses.

2. **Real-Time Data Visualization:**
   Implementation of real-time charts and dashboards using Chart.js or D3.js for better understanding of responses as they come in.

3. **Email and Notification System:**
   Addition of email alerts or push notifications to inform users about new survey responses or feedback updates.

4. **Multi-Language Support:**
   Enable multi-language functionality to allow users from different regions to create and respond to surveys in their preferred language.

5. **Cloud Deployment:**
   Hosting the application on cloud platforms such as AWS, Render, or Heroku to allow global access and scalability.

6. **Mobile Application Version:**
   Developing a mobile app version using React Native for better accessibility on smartphones.

7. **Data Security and Encryption:**
   Implementing advanced encryption techniques to protect sensitive survey responses and personal information.

By implementing these features, the Online Survey App can evolve into a powerful and intelligent feedback management platform that serves a wider audience with more advanced analytical capabilities.

REFERENCES

- React.js Documentation – https://react.dev/
- Tailwind CSS Documentation – https://tailwindcss.com/
- Vite Official Documentation – https://vitejs.dev/
- Spring Boot Framework – https://spring.io/projects/spring-boot
- Neon PostgreSQL Cloud Database – https://neon.tech/
- IntelliJ IDEA Documentation – https://www.jetbrains.com/idea/documentation/
- Axios HTTP Client – https://axios-http.com/
- React Router v6 Documentation – https://reactrouter.com/
- JSON Web Token (JWT) – https://jwt.io/introduction/
- Postman API Testing Tool – https://www.postman.com/

APPENDIX – USER MANUAL

1. System Requirements

Hardware Requirements:

- Processor: Intel i3 or higher
- RAM: Minimum 4 GB (8 GB recommended)
- Hard Disk: At least 2 GB of free space

Software Requirements:

- Operating System: Windows 10 / 11 or Linux / macOS
- IDEs and Tools:
  - IntelliJ IDEA (for backend)
  - Visual Studio Code (for frontend)
  - Node.js (for React environment)
  - Postman (for API testing)
- Database: Neon PostgreSQL (cloud-based)
- Web Browser: Chrome, Edge, or Firefox

2. Steps to Run the Application

A. Setting up the Backend (Spring Boot)

1. Open **IntelliJ IDEA** and import the Spring Boot project.
2. Configure the database connection in the **application.properties** file:
3. spring.datasource.url=jdbc:postgresql://your-neon-db-url
4. spring.datasource.username=your-username
5. spring.datasource.password=your-password
6. spring.jpa.hibernate.ddl-auto=update
7. Build and run the project using the command:
8. mvn spring-boot:run
9. Ensure that the backend server starts on **http://localhost:8080**.

B. Setting up the Frontend (React + Vite)

1. Open the frontend project folder in **Visual Studio Code**.
2. Install all required dependencies using the command:
3. npm install
4. Start the development server:
5. npm run dev
6. The app will run locally at **http://localhost:5173** (or another assigned port).

C. Connecting Frontend and Backend

1. Open the Axios configuration file in the frontend (for example: src/api/axios.js).
2. Set the base URL to point to your backend server:
3. const API_BASE_URL = "http://localhost:8080";
4. Save the file and restart the frontend server.
5. Test the connection by logging in or signing up; if successful, the app is correctly connected.

3. How to Use the Application

- **Login / Signup:**
  - Open the app in your browser.
  - Create a new account or log in using existing credentials.
- **Dashboard:**
  - After login, you will see all your created surveys listed on the dashboard.
  - You can also view the total number of responses for each survey.
- **Create Survey:**
  - Click the "Create New Survey" button.
  - Add a title and questions (text or multiple choice).
  - Click “Submit” to save your survey.
- **Share Survey:**
  - Once created, a unique survey link will be generated.
  - Copy and share this link with others to collect responses.
- **Take Survey (for participants):**
  - Open the shared link in any browser.
  - Fill in the answers and submit the survey.
- **View Analysis:**
  - Go back to your dashboard and click “View Analysis.”
  - You can see individual and aggregated responses displayed clearly.
- **Export Data:**

- Click the “Export CSV” option to download all survey responses for further use or analysis.

4. Troubleshooting

Issue	Possible Cause	Solution
Backend not starting	Incorrect database credentials	Check application.properties and Neon database URL
Frontend not loading	Missing dependencies	Run npm install again
Login not working	Backend API not connected	Verify backend server is running on port 8080
Data not visible	Database connection lost	Reconnect Neon PostgreSQL or restart backend
CORS error in browser	Frontend and backend ports differ	Enable CORS in Spring Boot configuration

5. Safety and Maintenance Instructions

- Always keep the database credentials private.
- Avoid modifying backend configurations without proper testing.
- Regularly update dependencies to keep the system secure.
- Backup survey data periodically from the Neon database.
- Use HTTPS for deployment to enhance data security.