Complex Problem-Solving Self-Assessment Form

1	Name of the Student	Mohammed Irfan
2	Roll Number	25951A6662
3	Branch and Section	CSE-(AI&ML) - A
4	Program	B. Tech
5	Course Name	Front-End Web Development
6	Course Code	ACSE04
7	Please tick (✓) relevant Engineering Competency (ECs) Profiles	

EC	Profiles	(✓)
EC 1	Ensures that all aspects of an engineering activity are soundly based on fundamental principles - by diagnosing, and taking appropriate action with data, calculations, results, proposals, processes, practices, and documented information that may be ill-founded, illogical, erroneous, unreliable or unrealistic requirements applicable to the engineering discipline	✓
EC 2	Have no obvious solution and require abstract thinking, originality in analysis to formulate suitable models.	✓
EC 3	Support sustainable development solutions by ensuring functional requirements, minimize environmental impact and optimize resource utilization throughout the life cycle, while balancing performance and cost effectiveness.	
EC 4	Competently addresses complex engineering problems which involve uncertainty, ambiguity, imprecise information and wide-ranging or conflicting technical, engineering and other issues.	✓
EC 5	Conceptualises alternative engineering approaches and evaluates potential outcomes against appropriate criteria to justify an optimal solution choice.	✓
EC 6	Identifies, quantifies, mitigates and manages technical, health, environmental, safety, economic and other contextual risks associated to seek achievable sustainable outcomes with engineering application in the designated engineering discipline.	

	EC 7	Involve the coordination of diverse resources (and for this purpose, resources include people, money, equipment, materials, information and technologies) in the timely delivery of outcomes	
	EC 8	Design and develop solution to complex engineering problem considering a very perspective and taking account of stakeholder views with widely varying needs.	✓
	EC 9	Meet all level, legal, regulatory, relevant standards and codes of practice, protect public health and safety in the course of all engineering activities.	

	EC 10	High level problems including many component parts or sub-problems, partitions problems, processes or systems into manageable elements for the purposes of analysis, modelling or design and then re-combines to form a whole, with the integrity and performance of the overall system as the top consideration.		✓
	EC 11	Undertake CPD activities to maintain and extend competences and enhance the ability to adapt to emerging technologies and the ever-changing nature of work.		✓
	EC 12	Recognize complexity and assess alternatives in light of competing requirements and incomplete knowledge. Require judgement in decision making in the course of all complex engineering activities.		✓
8	Please tick (✓) relevant Course Outcomes (COs) Covered			
	CO	**Course Outcomes**		**(✓)**
	CO 1	Describe language basics like alphabet, strings, grammars, productions, derivations, and Chomsky hierarchy, construct DFA, NFA, and conversion of NFA to DFA, Moore and Mealy machines and interpret differences between them.		✓
	CO 2	Recognize regular expressions, formulate, and build equivalent finite automata for various languages.		✓
	CO 3	Identify closure, and decision properties of the languages and prove the membership.		✓
	CO4	Demonstrate context-free grammars, check the ambiguity of the grammar, and design equivalent PDA to accept the context-free languages.		
	CO 5	Uses mathematical tools and abstract machine models to solve complex problems.		✓
	CO 6	Analyze and distinguish between decidable and undecidable problems.		✓
9	Course ELRV Video Lectures Viewed		**Number of Videos**	**Viewing time in Hours**
			-	-
10	Justify your understanding of WK1		-	

11	Justify your understanding of WK2 – WK9	-
12	How many WKs from WK2 to WK9 were implanted?	-
	Mention them	-

Date: 05-12-2025

Mohammed Irfan
Signature of the Student

COMPLEX ENGINEERING PROBLEM

A COURSE SIDE PROJECT

ON

Front-End Web Development

Mohammed Irfan

25951A6683

MarketYard ***A Project Report submitted in partial fulfillment of the***

requirements for the award of the degree of

Bachelor of Technology
in

CSE (Artificial Intelligence & Machine Learning)

By

Mohammed Irfan

25951A6662

Department of CSE (Artificial Intelligence & Machine Learning)

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad – 500 043, Telangana

November, 2025

DECLARATION

I certify that

a. The work contained in this report is original and has been done by me under the guidance of my supervisor (s).

b. The work has not been submitted to any other Institute for any degree or diploma.

c. I have followed the guidelines provided by the Institute for preparing the report.

d. I have conformed to the norms and guidelines given in the Code of Conduct of the Institute.

e. Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the report and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Mohammed Irfan

Place: Hyderabad **Signature of the Student**

Date: 05-12-2025

CERTIFICATE

This is to certify that the project report entitled MarketYard submitted by **Mohammed Irfan** to the Institute of Aeronautical Engineering, Hyderabad in partial fulfillment of the requirements for the award of the Degree Bachelor of Technology in **CSE - (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)** is a Bonafide record of work carried out by his guidance and supervision. The Contents of this report, in full or in parts, have not been submitted to any other Institute for the award of any Degree.

Supervisor **Head of the Department**

Date: 05-12-2025

Principal

APPROVAL SHEET

This project report entitled MarketYard submitted by **Mohammed Irfan** is approved for the award of the Degree Bachelor of Technology in Branch **CSE (Artificial Intelligence & Machine Learning).**

Examiner **Supervisor(s)**

Principal

Date: 05-12-2025

Place: Hyderabad

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crowns all efforts with success.

I am extremely grateful and express my profound gratitude and indebtedness to my project guide **Mr. Vidyasagar Vidapu, Assistant Professor, Department of CSE – (Artificial Intelligence and Machine Learning),** for his kind help and for giving me the necessary guidance and valuable suggestions for this project work.

I am grateful to **Dr. M. Purushotham Reddy**, **Professor** and **Head of the Department**, Department of **CSE (Artificial Intelligence & Machine Learning),** for extending his support to carry on this project work. I take this opportunity to express my deepest gratitude to one and all who directly or indirectly helped me in bringing this effort to present form.

I express my sincere gratitude to **Dr. L. V. Narasimha Prasad, Professor** and **Principal**

who has been a great source of information for my work.

I thank our college management and respected **Sri M. Rajashekar Reddy**, **Chairman**, **IARE, Dundigal** for providing me with the necessary infrastructure to conduct the project work.

I take this opportunity to express my deepest gratitude to one and all who directly or indirectly helped me in bringing this effort to present form.

ABSTRACT

Agricultural commodity prices are highly volatile and influenced by multiple factors such as climate conditions, seasonal patterns, transportation costs, government policies, and market demand. Farmers, traders, and agri-stakeholders often face challenges in understanding historical price behavior and identifying meaningful trends from raw market data. Traditional data sources are often presented in tabular or textual formats, which makes analysis time-consuming and difficult.

MarketYard is a front-end web-based dashboard designed to simplify agricultural market analysis by providing an interactive and visual approach to crop price exploration. The system allows users to browse price data by commodity, region, and date range, view historical trends using interactive charts, and compare multiple commodities or regions side-by-side. The dashboard also highlights sudden price spikes and drops and provides simple statistical summaries such as latest price and overall change.

The application is developed using HTML, CSS, JavaScript, and Chart.js. It uses mock datasets generated on the client side and does not require a backend server. MarketYard demonstrates how effective data visualization and decision-support tools can be built using lightweight web technologies, making market data more accessible and understandable.

CONTENTS

Name of Contents	**Page No.**
Title Page	I
Declaration	II
Certificate	III
Approval Sheet	IV
Acknowledgement	V
Abstract	VI
Contents	VII
Chapter 1- Introduction	12-13
1.1 Problem Statement	12
1.2 Introduction	12
1.3 Requirements	13
1.4 Prerequisites	13
1.5 Technologies used	13
Chapter 2 - Review of Relevant Literature	14
Chapter 3- Methodology	14
Chapter 4- Results and Discussions	14-15
Chapter 5- Conclusions and Future Scope	16
5.1 Conclusion	16
5.2 Future Scope	16
References	16

Chapter 1 – Introduction

MarketYard is a web-based crop price dashboard designed to help farmers, traders, and agricultural stakeholders understand historical commodity price movements across regions and time. It focuses on usability, clear visualizations, and lightweight implementation, relying on mock or static datasets generated in the browser rather than live APIs or databases.

The project addresses the difficulty many stakeholders face in interpreting raw tabular price data, which often comes from government bulletins or market feeds and is hard to analyze quickly. Interactive dashboards are widely used in agriculture and commodity trading to highlight trends, seasonal patterns, and anomalies, making complex datasets more actionable for non-technical users.

1.1 Problem Statement

Agricultural market data is often dispersed across portals and published as static tables or PDF bulletins, making it tedious for farmers and small traders to spot price trends, seasonal behavior, or sudden spikes across commodities and regions. This lack of accessible visualization can lead to sub-optimal decisions about when and where to sell produce.

The problem this project tackles is: how to provide a simple, browser-only dashboard that transforms historical crop price data into intuitive interactive charts, comparison views, and summary statistics, without requiring a heavy backend or complex installation. The solution must support basic exploration (by commodity, region, date range) and simple analytics (trend summaries, spike detection, CSV export) so that even a basic laptop or mobile browser is sufficient.

1.2 Introduction to MarketYard

MarketYard loads a synthetic dataset of three commodities (Wheat, Rice, Maize) across four regions (North, South, East, West) for the last 60 days, generated using a simple sinusoidal pattern with random noise to mimic realistic fluctuations. The interface then allows users to filter this mock dataset by commodity, region, date range, and time granularity (daily, weekly, monthly) and visualize the selected series in an interactive line chart using Chart.js.
Screenshot-2025-12-13-111838.jpg

In addition, the application offers a comparison mini-chart that shows other commodities for the same region, a “Compare commodity” feature that overlays a second commodity on the main chart, a “Recent movements” table that highlights large day-to-day spikes or drops, and a CSV export option for the currently viewed main series. These features together approximate the core behavior of production dashboards used in commodity analytics platforms while remaining fully front-end.

1.3 Requirements

The functional requirements include: browsing and searching price data by commodity, region, and date range; switching between daily, weekly, and monthly aggregated views; visualizing price trends in interactive charts; comparing multiple commodities for a given region; filtering and aggregating data; and exporting the current series as CSV. All of these are implemented via HTML form controls, DOM manipulation, and Chart.js line charts.Screenshot-2025-12-13-111838.jpg
Non-functional requirements emphasize responsiveness, clarity of visualization, and independence from a server backend. The layout is implemented using CSS grid and media queries to adapt between desktop and smaller screens, while the data is generated on the client and stored in JavaScript arrays to avoid network latency and simplify deployment on static hosting.

1.4 Prerequisites

To develop, run, or extend MarketYard, a basic understanding of HTML, CSS, and vanilla JavaScript is required, along with familiarity with Chart.js line charts and general concepts of data aggregation over time (daily vs weekly vs monthly). Knowledge of date handling in JavaScript and basic statistics (percentage change, moving averages) helps in understanding the logic that computes summaries and movements.youtubeScreenshot-2025-12-13-111838.jpg
On the user side, only a modern web browser with JavaScript enabled is needed, since all data and logic are executed client-side. Deployment can be as simple as hosting the HTML file on any static hosting service or opening it locally in a browser, making it accessible even in low-infrastructure environments.

1.5 Technologies Used

The project uses standard web technologies: HTML for structure, CSS for layout and styling, and JavaScript for data handling, filtering, aggregation, and DOM updates. Chart.js, a popular open-source charting library, is loaded via CDN and used to render the main price trend chart and the comparison chart as line charts with tooltips and legends. Screenshot-2025-12-13-111838.jpg
No external backend framework or database is used; instead, a mock dataset is generated in JavaScript using sine waves plus random noise for each commodity–region combination per day. The CSV export relies on the browser’s Blob and URL APIs to construct and download a CSV file corresponding to the current main chart series

Chapter 2 – Review of Relevant Literature

Several agricultural dashboards and crop information systems highlight the importance of visual interfaces for farmers to understand market and production patterns. Academic and industry projects on crop dashboards typically include time-series plots, filters for crop and region, and summary metrics to support decision-making in planting and selling strategies.

Commodity price dashboards in finance and agriculture often rely on more advanced stacks (e.g., Dash, Flask, or BI tools) but share common features with MarketYard, such as interactive time-series charts, date-range selection, and comparison between commodities. These systems confirm that visualizing multi-series price data and providing simple statistics like price changes and correlations can substantially improve interpretability for users.

Front-end-only dashboards are increasingly common for teaching, prototyping, and hackathons because they can simulate full analytics experiences using static or mock data. Guidance on project reports for such systems recommends including an introduction, literature review, methodology, results, and conclusions, with emphasis on clear explanation of visualizations and user interaction design.

Chapter 3 – Methodology

The overall methodology for MarketYard follows a front-end-only architecture, where data generation, filtering, analytics, and rendering all occur in the browser. A mock dataset for 60 days is created by iterating over dates and over the list of commodities and regions, computing a base price plus a periodic component and random noise to simulate seasonal and stochastic variation. Once generated, the dataset is stored in an array of objects with fields for date, commodity, region, and price, which simplifies filtering by value matching and date comparisons. Aggregation functions compute daily, weekly, or monthly series by grouping records into buckets (using week or month keys) and averaging prices in each group to create smoother trend lines.

Chapter 4 – Results and Discussion

The resulting interface presents a dark-themed dashboard with a sticky header containing the MarketYard logo and control panel, a main panel for price trends, and a side panel for comparison and recent movements. The layout is implemented with CSS grid so that on wide screens the main chart and side widgets appear side by side, while on smaller screens they stack vertically for better readability.

Users can interactively select commodity, region, aggregation view, and date range, then click “Apply” to refresh the main chart and summary statistics. The main chart shows the aggregated time series as a smooth line with tooltips and legend entries, and a textual summary above it highlights the selected combination and the latest price and change, supporting quick decision-making at a glance

4.1 Visualization Outcomes

The visualization technique chosen is a line chart, which is well suited to time-series data and supports comparing multiple series by overlaying lines of different colors. Chart.js is configured to use category scales for dates, with reduced tick counts to avoid clutter, and light-colored axes and legend labels to match the dark background and maintain readability

The comparison chart in the side panel automatically plots the other commodities (excluding the currently selected base commodity) for the same region and date range, using distinct colors so that users can immediately see relative levels and trend shapes. In addition, the “Compare commodity” button allows manually overlaying a chosen second commodity directly on the main chart, making side-by-side comparison more prominent.

4.2 Functional Evaluation

All specified functional requirements are addressed: commodity and region selection via dropdowns; date-range filtering via native date inputs; daily, weekly, and monthly views via a view selector; interactive line charts with tooltips and legends; comparison capabilities; filtering and aggregation; and CSV export of the current main series. The CSV export feature reads the active dataset from the main chart, builds a two-column CSV (date, price), and triggers a download with a filename derived from the summary label.

The “Recent movements” table adds a layer of anomaly detection by surfacing large price changes across commodities and regions for the last 10 days, tagged as “Spike” or “Drop” with the percentage change. This helps users quickly identify potential opportunities or risks without manually scanning the graphs, similar to alert widgets in professional market dashboards.

4.3 Discussion

From a usability standpoint, the dashboard keeps controls compact and uses concise labels, but there is room to enhance accessibility with tooltips, icons, and clearer color coding for color-blind users. The reliance on a single page and minimal controls makes the learning curve low, which is advantageous for farmers and small traders who may not be familiar with complex analytics tools

From a technical perspective, the fully front-end architecture ensures easy deployment and low maintenance but limits scalability for very large or real-time datasets. For educational and demonstration purposes, however, this architecture is ideal because it avoids backend setup, demonstrates core data visualization concepts, and can be extended later to consume real APIs or datasets.

Chapter 5 – Conclusions and Future Scope

5.1 Conclusion

MarketYard demonstrates how a purely front-end application can deliver a functional crop price dashboard with interactive time-series charts, comparison views, anomaly highlighting, and CSV export using only HTML, CSS, JavaScript, and Chart.js. It addresses the problem of inaccessible tabular price data by turning it into intuitive visualizations and summaries that are easier for farmers and traders to interpret

The project validates that mock datasets, when designed carefully to reflect realistic price dynamics, are sufficient for exploring interface design, data aggregation, and visualization patterns relevant to agricultural markets. It also shows that such dashboards can be packaged as lightweight static sites suitable for labs, coursework, and early prototypes of larger market information systems.

5.2 Future Scope

Several enhancements can extend MarketYard toward a more production-ready tool. One direction is integrating real market price feeds or open datasets for agricultural commodities, potentially via APIs from government portals or commodity exchanges, along with caching and pagination to handle volume efficiently.

Another direction is adding optional modules such as a regional map view using a mock or real map API, configurable threshold alerts with notifications, moving-average trend indicators, and CSV import to let users load their own price data. Moving the logic to a component-based framework like React and connecting to a backend service would also support multi-user customization, authentication, and persistence of user preferences.

References

The report structure (chapters, headings such as Introduction, Methodology, Results, Conclusion, and Future Scope) follows standard guidelines for academic and project reports on web applications and dashboards