# TABLE OF CONTENT

***TOPIC PAGE:***

# ACKNOWLEDGEMENT

I would like to express my profound gratitude to **Ms. Shagufta Siddiqui** ma'am sir of BCA department for her contributions to the completion of my project titled Rainwater Harvesting Planner App. Your useful advice and suggestions were really helpful to me during the project's completion. In this aspect, I am eternally grateful to you. I would also like to thank all of the other supporting personnel who assisted me by supplying the equipment that was essential and vital, without which I would not have been able to perform efficiently on this project.

I would also want to thank Shri Ramswaroop Memorial College Of Management for accepting my project in my desired field of expertise. I am grateful to the college administration for providing me with such a significant chance. I'd also like to thank my friends and parents for their support and encouragement as I worked on this assignment.

# PREFACE

In the preparation of the project , I have precisely demarcated all the important points. I have made best possible effort to remove all the errors. The work presented here is the result of several months of research, planning, and execution. It required an in-depth understanding of Web Development with MERN and the application of various methodologies and techniques learned throughout the course. The successful completion of this project was made possible through the guidance, support, and encouragement of several individuals.

I would like to express my sincere gratitude to my project supervisor, **Ms. Shagufta Siddiqui maam** for their continuous support, insightful advice, and patience throughout the development of this project. Their expertise and feedback were invaluable in shaping the direction and outcome of this work.

I am also grateful to the faculty member who provided assistance and resources that were crucial to the success of this project. Additionally, I extend my appreciation to my peers for their collaboration and constructive discussions, which enriched my understanding and helped me refine my ideas.

Finally, I would like to acknowledge the unwavering support of my family and friends, whose encouragement kept me motivated during challenging times.

I hope that this project contributes meaningfully to the academic discourse in cloud computing and serves as a foundation for further research and exploration in this area.

# DECLARATION

I Divya Rani Sharma a student of BCA department at Shri Ramswaroop Memorial College Of Management hereby declare that the project titled **Rainwater Harvesting Planner App** submitted is my original work and has not been submitted earlier to any other institution or university for the award of any degree or diploma.

I further declare that all sources of information, data, and material used in this project have been duly acknowledged in the bibliography or references section. I have followed the prescribed ethical guidelines in the collection and presentation of data and have taken due care to ensure that no information presented is plagiarized.

This work was carried out under the guidance and supervision of **Ms**. **Shagufta Siddiqui ma'am** I accept full responsibility for any errors or omissions in this project.

Divya Rani Sharma

2312044050056

BCA Department

# CHAPTER 1

# 1 INTRODUCTION

## *1.1 BACKGROUND*

Rainwater harvesting is an essential sustainable solution for water conservation. With increasing urbanization and climate uncertainty, managing rainfall effectively has become a pressing need. The Rainwater Harvesting Planner App is a web-based system built using the MERN stack (MongoDB, Express.js, React.js, Node.js), which helps users predict rainfall in their area, calculate potential water collection, and receive suggestions on how to reuse or store harvested rainwater effectively. Rainwater harvesting is a sustainable technique that involves collecting, storing, and utilizing rainwater for various purposes such as irrigation, household use, or groundwater recharge. With advancements in web technologies and data analytics, it has become possible to build smart systems that not only spread awareness but also help users actively manage their rainwater harvesting potential. A Rainwater Harvesting Planner App can play a vital role by predicting local weather and rainfall patterns, calculating rainwater availability, and providing personalized suggestions for efficient rainwater use.

## *1.2 OBJECTIVE*

A **Rainwater Harvesting Planner App** aims to promote sustainable water management by providing a centralized digital platform for planning, tracking, and optimizing the collection and reuse of rainwater. This application is designed to empower individuals, households, and communities to make informed decisions based on real-time weather forecasts, rainfall predictions, and water usage analytics.

The key objectives of the Rainwater Harvesting Planner App are:

- Analyze rainfall patterns and forecast weather conditions to help users plan rainwater collection.
- Calculate the potential rainwater harvesting capacity using parameters such as rooftop area, tank size, and local rainfall data.
- Assist users in tracking water storage levels and managing usage efficiently to avoid overflow or wastage.
- Provide real-time notifications and alerts for upcoming rainfall events or storage issues, enabling timely action.
- Promote reuse of collected rainwater through suggestions for household or gardening applications, thereby reducing dependence on external water sources.

# *1.3 PURPOSE AND SCOPE*

## *1.3.1 PURPOSE*

The purpose of the Rainwater Harvesting Planner App is to promote sustainable and efficient use of rainwater by providing users with a smart, interactive platform that assists in planning, monitoring, and optimizing rainwater collection and reuse. This application addresses the growing concern of water scarcity and inefficient water usage by leveraging modern web technologies to deliver accurate rainfall forecasts, calculate harvestable rainwater, and offer suggestions for reuse. A rainwater harvesting planner app aims to promote water conservation by helping users design and implement effective rainwater harvesting systems. It scopes to assess the feasibility, optimize system design, and provide guidance on maintenance, ultimately reducing reliance on traditional water sources and mitigating water scarcity.

## *1.3.2 SCOPE*

The scope of the proposed **Rainwater Harvesting Planner App** defines the range of features, functionalities, and goals aimed at helping users plan, track, and optimize the use of rainwater through a digital platform. Below is a structured outline of the scope:

- **Rainwater Estimation and Calculation**: Enable users to input parameters such as rooftop area and local rainfall data to calculate the estimated volume of harvestable rainwater.
- **Real-Time Weather Forecasting**: Integrate weather APIs to provide up-to-date forecasts, rainfall predictions, and alerts to help users plan water collection accordingly.
- **Water Storage Simulation**: Allow users to manage and visualize their storage tank levels and get notifications for overflow or low water levels.
- **Reuse Planning and Suggestions**: Offer practical and personalized recommendations for reusing collected rainwater in activities such as gardening, cleaning, or flushing.
- **User Dashboard and Analytics**: Offer a secure and user-friendly dashboard where users can track rainwater usage history, view analytics, and manage their settings.
- **Cross-Platform Accessibility**: Design a responsive web application that works seamlessly across desktops, tablets, and mobile devices for easy access from any location.
- **Secure Authentication and Data Handling**: Ensure secure login, data privacy, and appropriate user-level access to prevent unauthorized usage or data breaches.

# CHAPTER 2

## 2.SURVAYS AND TECHNOLOGIES

The MERN stack is a popular full-stack JavaScript framework that includes MongoDB, Express.js, React.js, and Node.js**.** It provides a powerful and efficient development environment for building dynamic and data-driven web applications. In this project, the MERN stack is used to develop the Rainwater Harvesting Planner App**,** enabling real-time interaction, data storage, and a seamless user experience.

### ◆ MongoDB (Database Layer)

- Stores user data, including registration and login details securely.
- Maintains records of each user's rainwater harvesting configurations such as rooftop size and tank capacity.
- Saves historical rainfall data and analytics to help users track water collection over time.

---

### ◆ Express.js (Backend Framework)

- Acts as the server-side framework that connects the React frontend with the MongoDB database.
- Handles all RESTful API endpoints, such as fetching weather data, performing rainwater calculations, and processing user actions.
- Ensures secure, efficient, and organized communication between the client interface and the backend server.

---

### ◆ React.js (Frontend Framework)

- Builds a dynamic, responsive, and interactive user interface that enhances the user experience.
- Displays real-time weather data, rainfall forecasts, rain probability charts, and water usage statistics.
- Enables the UI to update instantly when users input data or when live weather updates are received, ensuring a smooth and modern interface across devices.

---

## ◆ Node.js (Runtime Environment)

- Provides the runtime environment to execute JavaScript code on the server side, running the backend logic.
- Interfaces with external APIs like WeatherAPI.com to fetch live weather forecasts and rainfall data.
- Handles real-time data processing tasks such as calculating rainwater volume, while ensuring fast and scalable performance for all user requests and background operations.

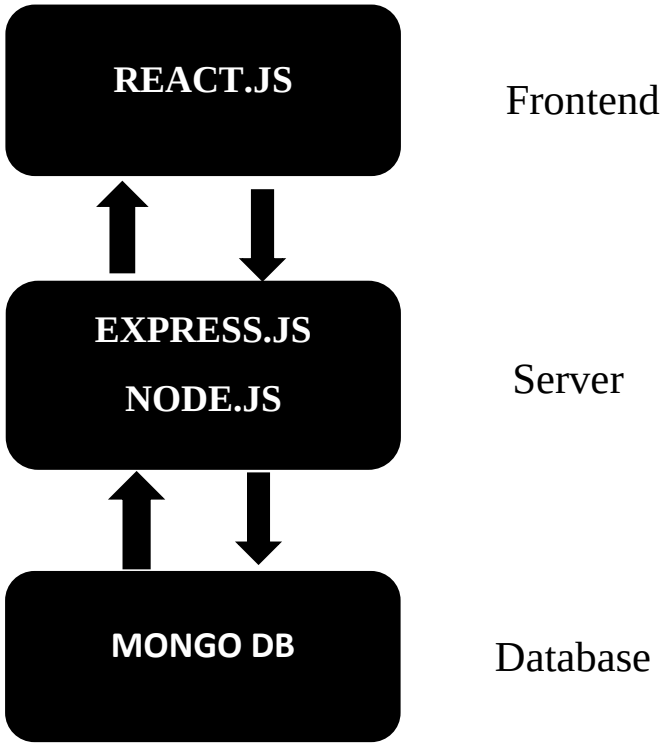



Fig: Functioning of MERN

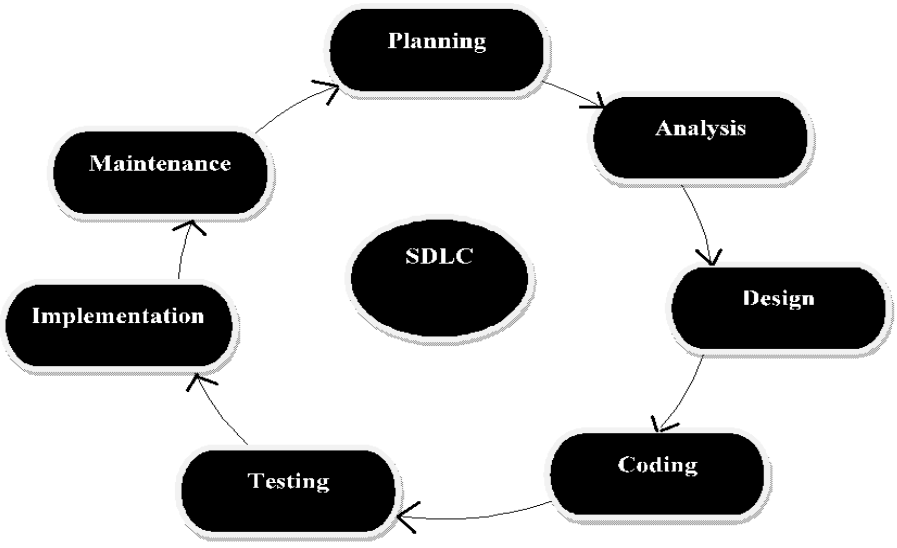
# CHAPTER 3

## 3.1 SOFTWARE DEVEPOLMENT LIFE CYCLE

The software development lifecycle (SDLC) is the cost-effective and time-efficient process that development teams use to design and build high-quality software. The goal of SDLC is to minimize project risks through forward planning so that software meets customer expectations during production and beyond. The software development lifecycle (SDLC) methodology provides a systematic management framework with specific deliverables at every stage of the software development process.. A life cycle model represents all the methods required to make a software product transit through its life cycle stages. It also captures the structure in which these methods are to be undertaken. In other words, a life cycle model maps the various activities performed on a software product from its inception to retirement.. It provides a systematic process for planning, creating, testing, and deploying software. The SDLC aims to produce high quality software that meets or exceeds customer expectations, is delivered on time, and is within budget.

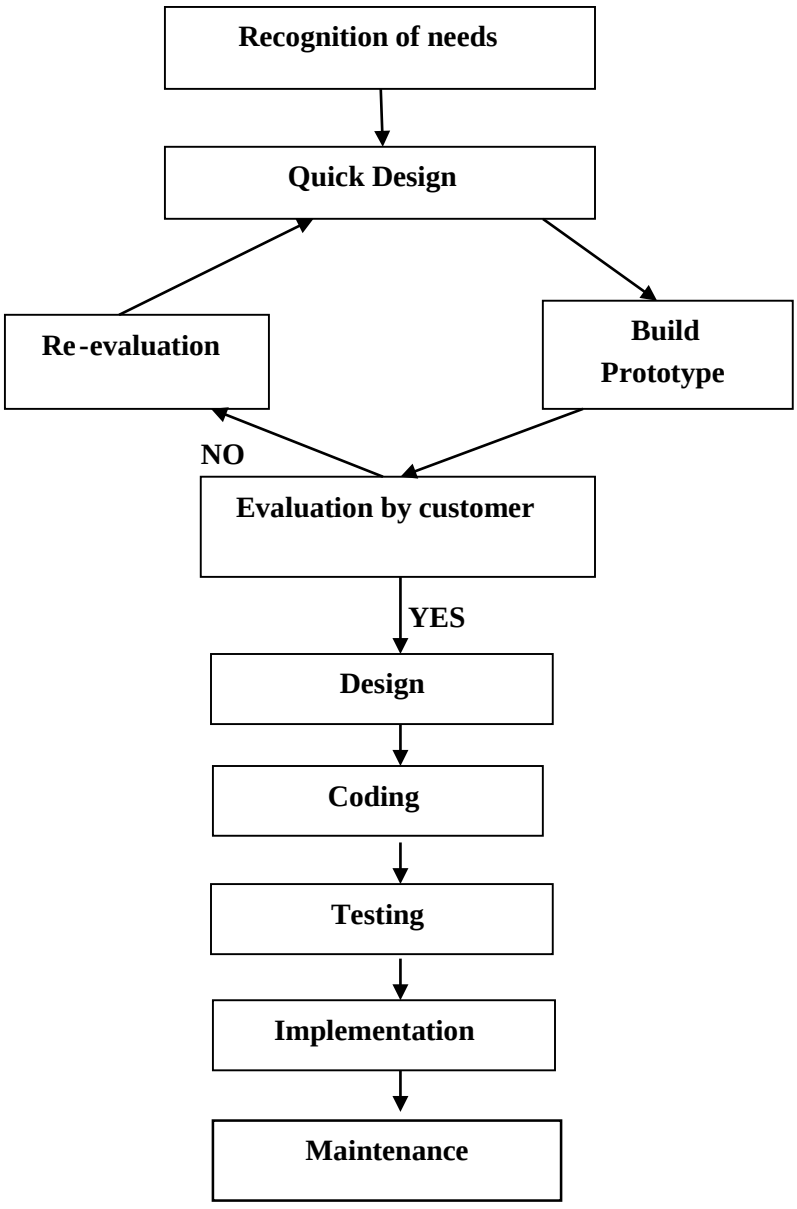
Software Development Life Cycle comprises of seven different stages. Following are the different activities to be considered while defining SDLC :

- Planning
- Analysis
- Design
- Coding
- Testing
- Implementation
- Maintainence



# 3.1 PROTOTYPE MODEL

The **Prototype Model** is a software development methodology where a preliminary version of the software, called a prototype, is developed to understand and refine the requirements of the final product. In this model, a prototype of the end product is first developed, tested, and refined as per customer feedback repeatedly till a final acceptable prototype is achieved which forms the basis for developing the final product. Provides a tangible representation of the software early in the development process. Involves users actively in the development process by demonstrating prototypes and gathering feedback. Helps in refining and clarifying requirements that are often unclear or evolving. Provides a clearer understanding of requirements through interactive prototypes, leading to more accurate final specifications. Allows for changes and refinements based on user feedback without extensive rework.



# CHAPTER 4

# 4. REQUIREMENT ANALYSIS

## *4.1 PROBLEM DEFINITION*

In today's world, water scarcity is becoming a critical issue due to rapid urbanization, climate change, and inefficient water management practices. While rainwater harvesting presents an effective and sustainable solution, many individuals and communities lack the knowledge, tools, and planning resources needed to implement it effectively. Traditional approaches to rainwater harvesting are manual and disconnected from modern technological advancements, which limits their reach and efficiency. Users often struggle with understanding the amount of rainwater they can collect based on their location, rooftop area, and rainfall data. There is also a lack of systems that can dynamically predict rainfall, track storage levels, and provide actionable suggestions for the reuse of rainwater. To address these issues, a smart, web-based Rainwater Harvesting Planner App is proposed. This system uses real-time weather data and user input to help plan and manage rainwater harvesting efficiently. It aims to bridge the gap between technology and water conservation by providing a digital tool that is accessible, informative, and highly interactive. The absence of such a system not only leads to missed opportunities for rainwater collection but also contributes to poor water management in both urban and rural areas.

## *4.2 FEASIBILITY STUDY*

A feasibility study is conducted to evaluate whether the proposed Rainwater Harvesting Planner App is viable in terms of technology, cost, operations, and legality. This assessment helps ensure that the system can be successfully developed and deployed using current tools and within practical constraints.

The feasibility study covers the following aspects:

- ✓ **Technical Feasibility**:
  The app is feasible using the MERN stack and weather APIs, which support real-time and full-stack development effectively.
- ✓ **Operational Feasibility**:
  The system is user-friendly, responsive, and easy to operate across various devices with minimal effort.
- ✓ **Financial Feasibility**:
  Development costs are low due to the use of open-source tools and affordable API services, making it budget-friendly.
- ✓ **Legal Feasibility**:
  The system follows standard data privacy norms and uses publicly available data, ensuring legal compliance.

## *4.3 PLANNING AND SCHEDULING*

Planning and Scheduling involves outlining the project's tasks, defining milestones, and setting timelines to ensure timely and successful development and deployment. This phase is crucial for coordinating resources, managing risks, and ensuring that the project meets its objectives. Below is a structured approach to planning and scheduling for a Rainwater Harvesting Planner App:

| S.No | Phases of project | No. of days | Starting | Ending |
|---|---|---|---|---|
| 1 | Project Initiation | 2 day | 08/7/24 | 10/7/24 |
| 2 | Requirement Analysis | 3 days | 11/7/24 | 14/7/24 |
| 3 | System Architecture Design | 2 days | 14/7/24 | 16/7/24 |
| 4 | Database Design | 4 days | 17/7/24 | 21/7/24 |
| 5 | Database Implementation | 4 days | 22/7/24 | 26/7/24 |
| 6 | Integration Phase | 2 days | 27/7/24 | 29/7/24 |
| 7 | Testing Phase | 3 days | 30/7/24 | 2/8/24 |
| 8 | Bug and Refinement | 3 days | 3/8/24 | 6/8/24 |
| 9 | Deployment Preparation | 3 days | 7/8/24 | 10/8//24 |
| 10 | Documentation Phase | 2 days | 11/8/24 | 13/8/24 |
| 11 | Post- Deployment Phase | 2 days | 14/8/24 | 16/8/24 |
| 12 | Development Complete | 2 days | 17/8/24 | 19/8/24 |
| 13 | System Deployed | 4 days | 20/8/24 | 24/8/24 |
| 14 | Final Project Review | 2 days | 24/8/24 | 26/8/24 |

## *4.4. REQUIREMENT SPECIFICATION*

❖ **SOFTWARE REQUIREMENT:**

- **Operating System**: Windows 11
- **Frontend**: React.js
- **Backend**: Node.js with Express.js
- **Database**: MongoDB
- **API Integration**: WeatherAPI (or similar) for real-time weather data

❖ **HARDWARE REQUIREMENT:**

- **Processor**: Intel Core i7
- **RAM**: Minimum 8 GB
- **Storage**: Minimum 100 GB of available storage
- **Input Devices**: Standard Keyboard and Mouse

## *4.5 . PREMINARY PRODUCT DESCRIPTION*

1. **Login**
   The application begins with a secure login system where users authenticate using their credentials. This ensures authorized access to features like location selection, weather data viewing, and rainwater harvesting suggestions. It helps in maintaining personalized settings and tracking user interactions with the app.
2. **Location-Based Weather Forecasting**
   The system provides live and accurate weather data based on the selected location. Users can choose from predefined locations (e.g., nearby Lucknow) and instantly receive current conditions, hourly rain predictions, and a 7-day forecast using WeatherAPI.
3. **Rain Probability Table**
   The app displays rain percentage in a tabular format for every hour, helping users understand when rainfall is most likely to occur. This aids in scheduling water collection or usage.
4. **Rainwater Harvesting Suggestions**
   Based on upcoming weather forecasts and rain probabilities, the app provides intelligent, dynamic suggestions on how to collect, store, and reuse rainwater efficiently.
5. **Live Rain Probability Chart**
   A graphical representation (chart) of rain likelihood throughout the day provides a visual, intuitive understanding of rainfall trends to better plan harvesting activities.
6. **Responsive & Interactive UI**
   The user interface is built using React.js for responsiveness and real-time updates. It is optimized for both desktop and mobile devices, making it accessible and user-friendly.

7. **Location Dropdown**
   Instead of manual input, users can select from a dropdown of predefined nearby locations, improving ease of use and reducing errors in data retrieval.
8. **Real-Time Data Fetching**
   The system dynamically pulls weather data from WeatherAPI.com using asynchronous calls, ensuring users always receive the most up-to-date forecast information.
9. **Environmental Awareness and Education**
   The app encourages sustainable practices by educating users on how much rain they can potentially collect and offering tips for responsible water reuse.

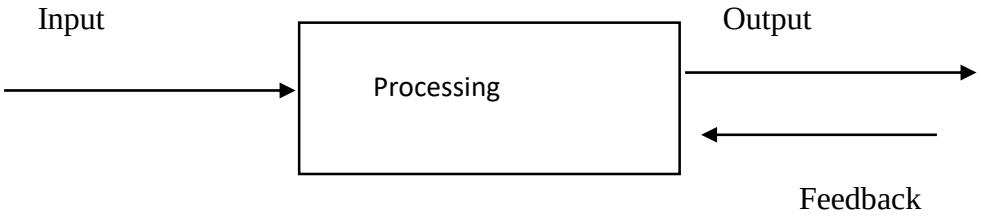
# 4.6 CONCEPTUAL MODELS

## *4.6.1 SYSTEM ARCHITECTURE DESIGN*

### *1.2 Defining System*

Defining a system provides a comprehensive blueprint that guides its development, implementation, and operation, ensuring that all aspects are clearly understood and aligned with its intended purpose. Defining a system using the Input, Processing, Output, and Feedback model provides a clear understanding of how the system operates, how it transforms data, and how it responds to user interactions and performance metrics.

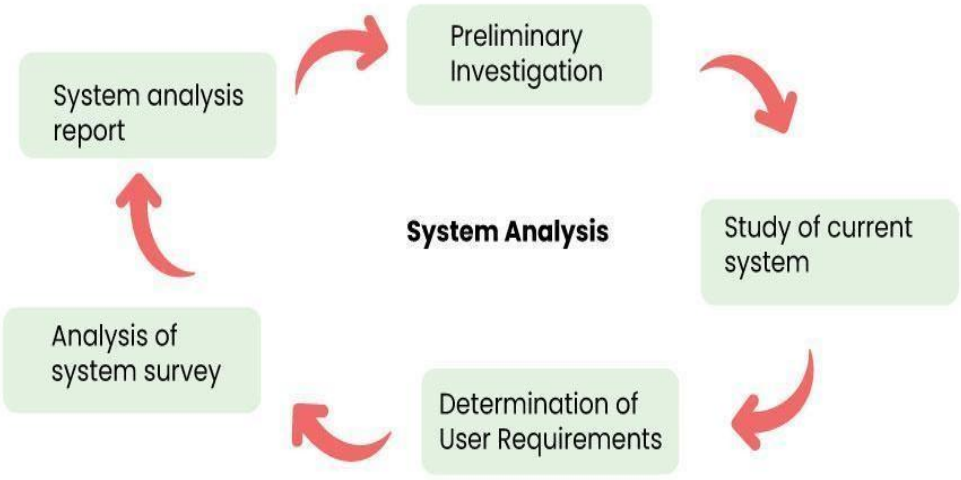
- **Input:** The data or resources that are entered into the system for processing. For example, in a document collaboration system, inputs could include user-uploaded files or comments.
- **Processing:** The operations or transformations performed on the inputs to achieve desired results. This includes activities like editing documents or managing versions.
- **Output:** The results produced by the system after processing the inputs. Outputs in a document collaboration system might include updated documents or user notifications.
- **Feedback:** Information returned to the system or users about its performance, used for evaluation and improvement. In this context, feedback could include user comments or error reports, helping to refine the system's functionality.




### *1.3 System Analysis*

System analysis is a critical phase in the development of a system. It involves breaking down a complex system into its constituent parts to understand its structure, function, and behavior. It is the

very first step in any system development and the critical phase where developers come together to understand the problem, needs, and objectives of the project.

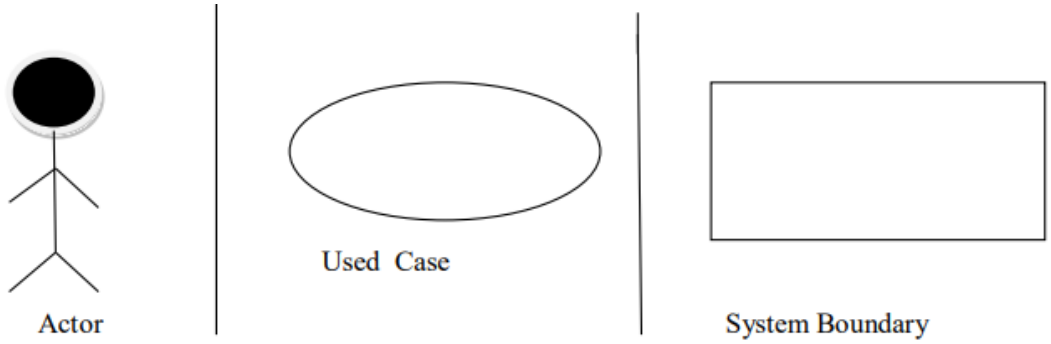


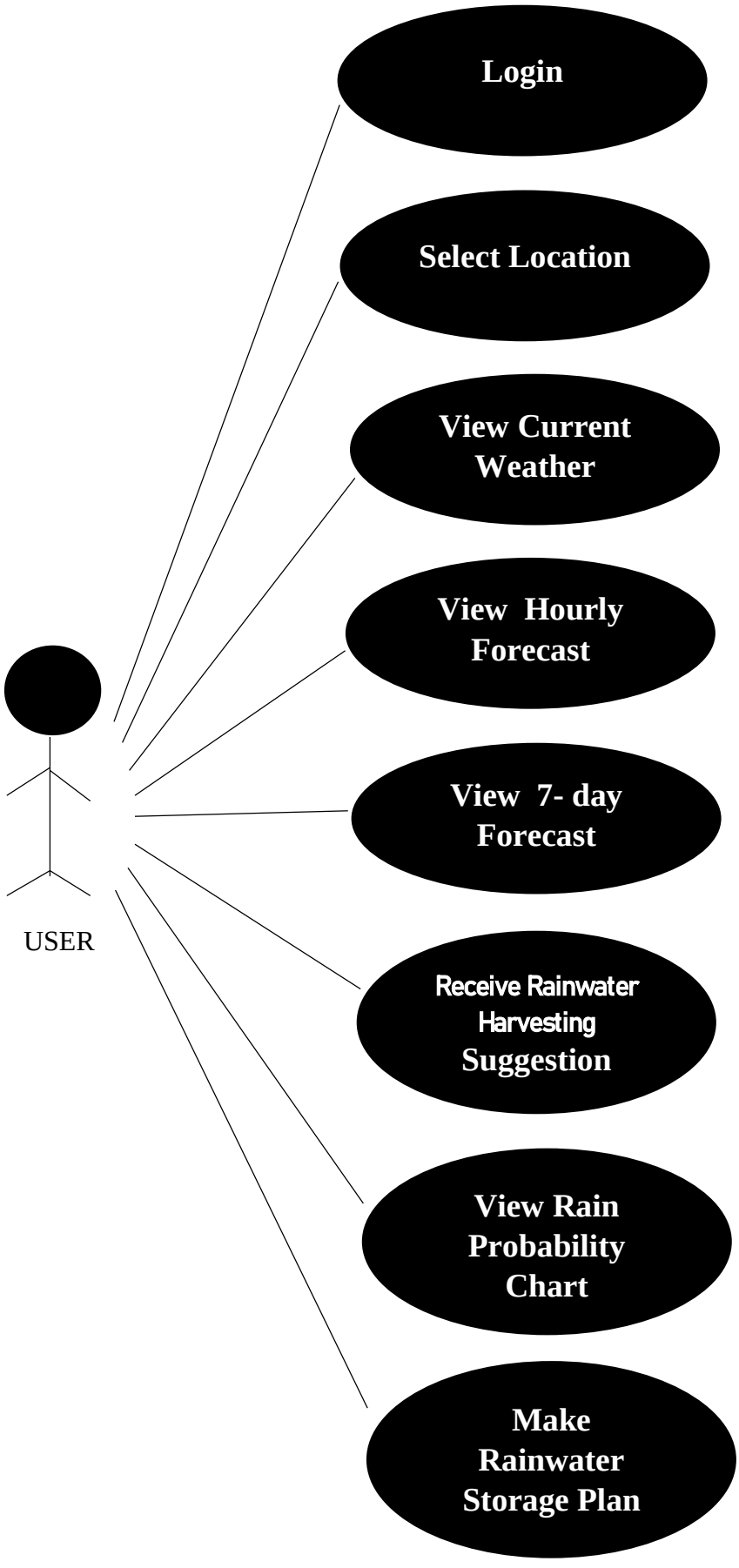

## USE CASE DIAGRAM

A use case diagram is a visual representation of the interactions between a system and its external actors. It provides a high-level overview of the system's functionality and the actions that users can perform. It helps to capture the functional requirements of a system by showing what the system does (its use cases) and which external entities (actors) interact with it

Components of a Use Case Diagram:

a. **Actors:** The entities that interact with the system. They can be people, organizations, or other systems.
b. **Use Cases:** The actions or services that the system provides to its actors. Each use case represents a specific goal or task that an actor can achieve.
**c. Relationships:** The connections between actors and use cases. These relationships can be include, extend or generalize.




Login
Select Location
View Current Weather
View Hourly Forecast
View 7- day Forecast
Receive Rainwater Harvesting Suggestion
View Rain Probability Chart
Make Rainwater Storage Plan
USER

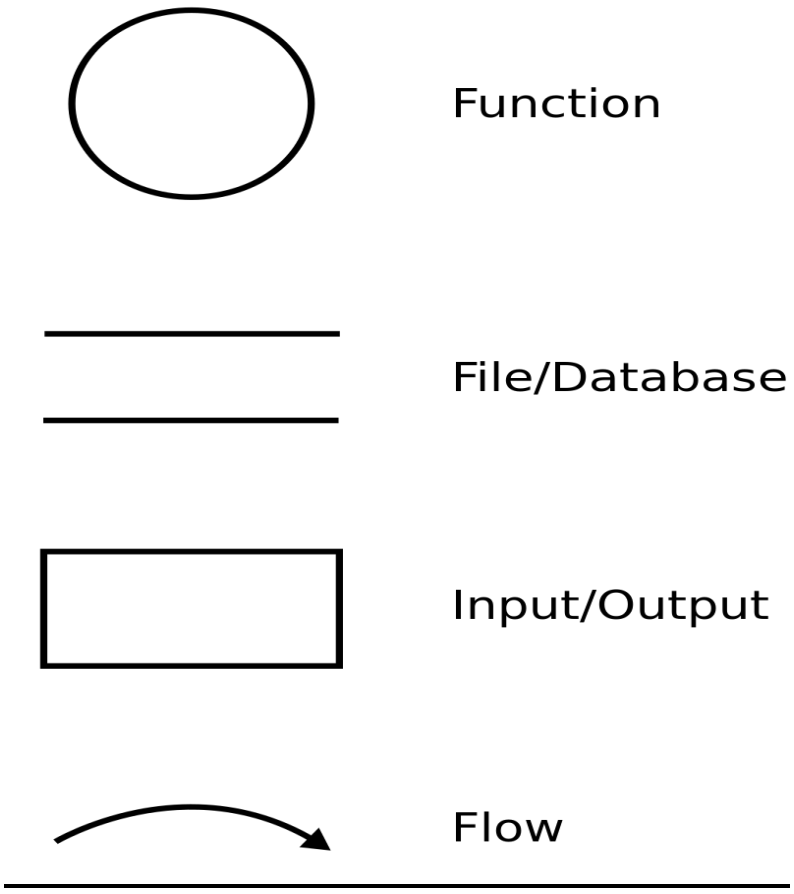
# DATA FLOW DIAGRAM

**Data Flow Diagrams (DFDs)** are graphical tools used to represent the flow of data through a system. They provide a high-level view of how data enters the system, is processed, and is output. It shows how data enters the system, how it's processed, and how it exits the system. DFDs are useful for understanding the structure of a system and how various components interact with each other.

**Components of a DFD:**

- **Process:** A circle or rectangle representing a function or activity that transforms data.
- **Data Flow:** An arrow indicating the direction of data movement between processes or data stores.
- **Data Store:** A rectangle representing a repository of data.
- **External Entity:** A square representing an external source or destination of data

- **SYMBOLS:**



## Zero Level DFD

It is also known as a context diagram. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.

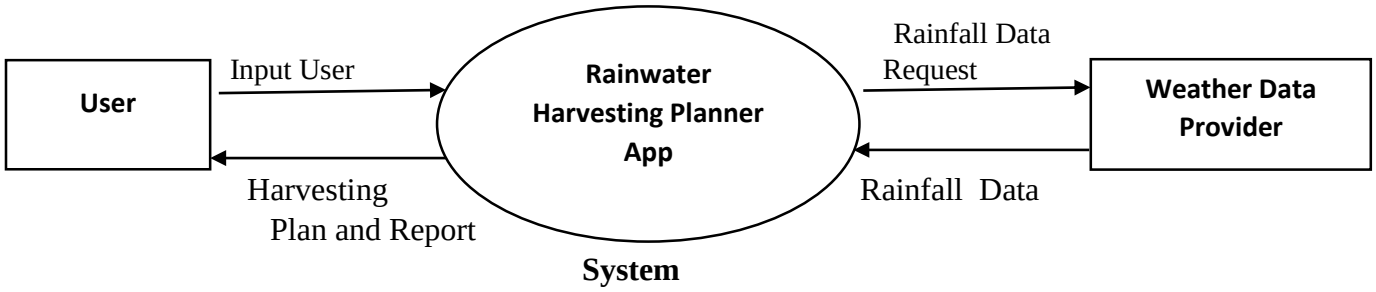


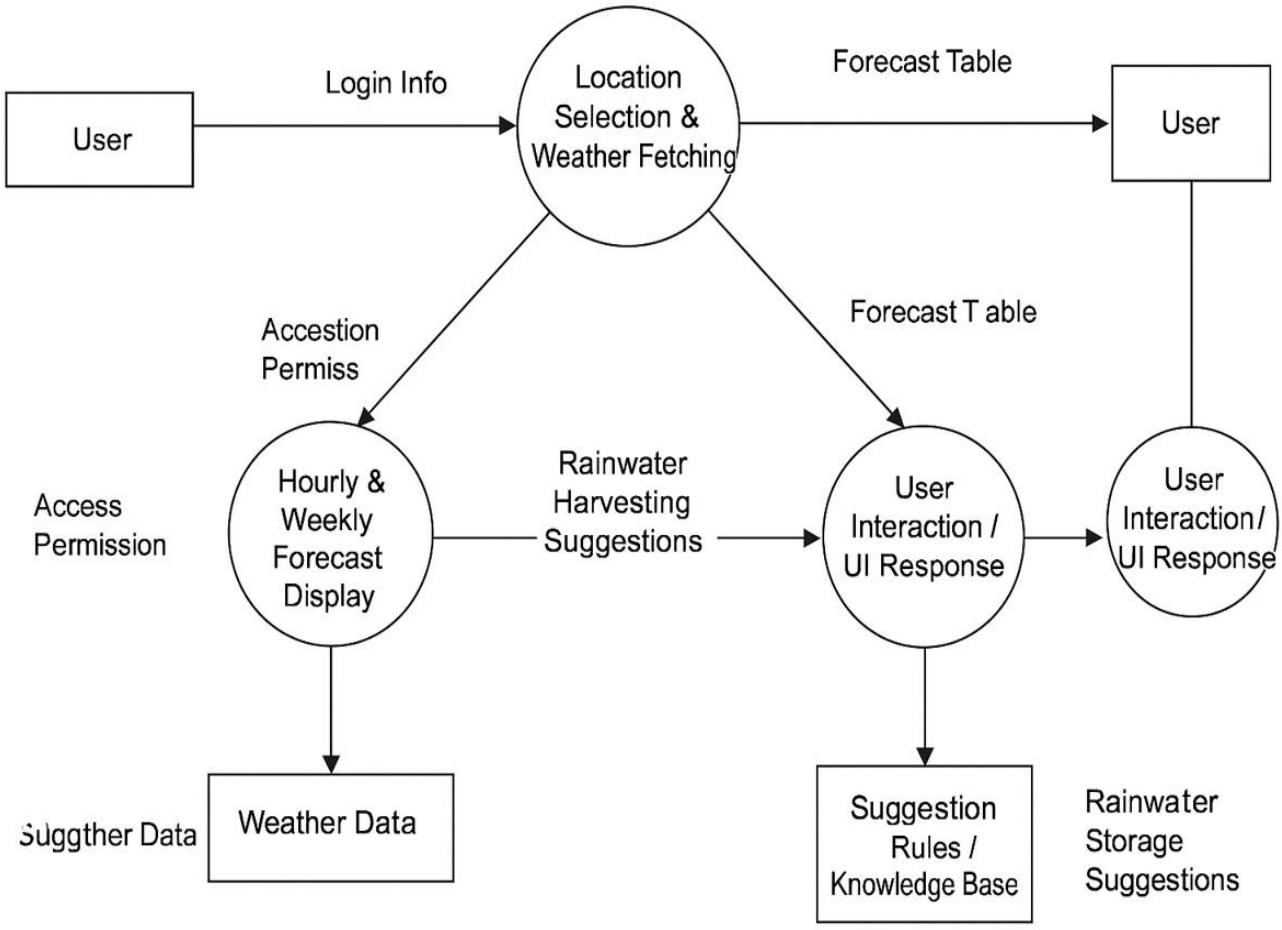

**Fig: Zero Level Data Flow Diagram**

## First Level DFD

This level provides a more detailed view of the system by breaking down the major processes identified in the level 0 DFD into sub-processes. Each sub-process is depicted as a separate process on the level 1 DFD. The data flows and data stores associated with each sub-process are also shown. This diagram breaks down the single process from the context diagram into subprocesses, showing more detailed data flows within the system.

### Major Points:

- ✓ **High-level view**: Focuses on the major processes and data flows within the system.
- ✓ **Limited detail**: Does not show the internal workings of processes.
- ✓ **Clear boundaries:** Defines the system's boundaries and interactions with external entities.


User
Login Info
Location Selection & Weather Fetching
Forecast Table
User
Accestion Permiss
Forecast T able
Access Permission
Hourly & Weekly Forecast Display
Rainwater Harvesting Suggestions
User Interaction / UI Response
User Interaction/ UI Response
Suggther Data
Weather Data
Suggestion Rules / Knowledge Base
Rainwater Storage Suggestions

# E R DIAGRAM

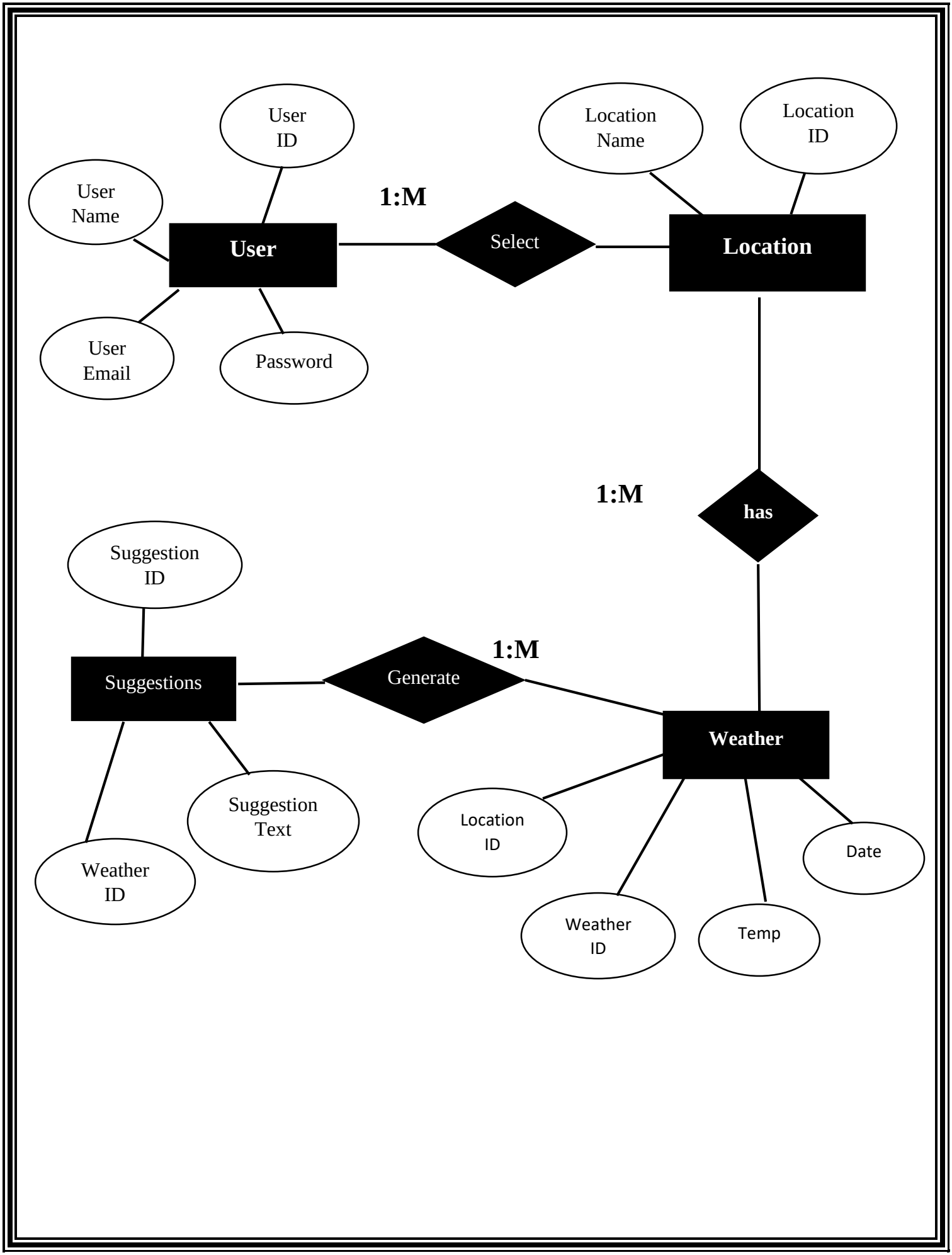
**Entity-Relationship (ER) Diagrams** are a graphical tool used to model data and their relationships in a database. They provide a visual representation of entities (objects or concepts), attributes (properties of entities), and relationships between entities.

An ER diagram can help businesses document existing databases and thereby troubleshoot logic or deployment problems or spot inefficiencies and help improve processes when a business wants to undertake business process re-engineering. ERDs can also be used to design and model new databases and make sure that engineers can identify any logic or design flaws before they're implemented in production.

- Document an existing database structure
- Debug, troubleshoot, and analyze
- Design a new database
- Gather design requirements
- Business process re-engineering

**Components of an ER Diagram:**

- **Entity:** A person, place, thing, or concept that can be uniquely identified. It is represented by a rectangle.
- **Attribute:** A property or characteristic of an entity. It is represented by an oval or circle
- **Relationship:** The association between two or more entities. It is represented by a diamond
- **Cardinality:** The number of instances of one entity that can be related to another entity. It is represented by a notation near the relationship diamond (e.g., 1:1, 1:M, M:M).


User ID
User Name
User
1:M
Select
Location Name
Location ID
Location
User Email
Password
1:M
has
Suggestion ID
Suggestions
Generate
1:M
Weather
Suggestion Text
Weather ID
Location ID
Date
Weather ID
Temp

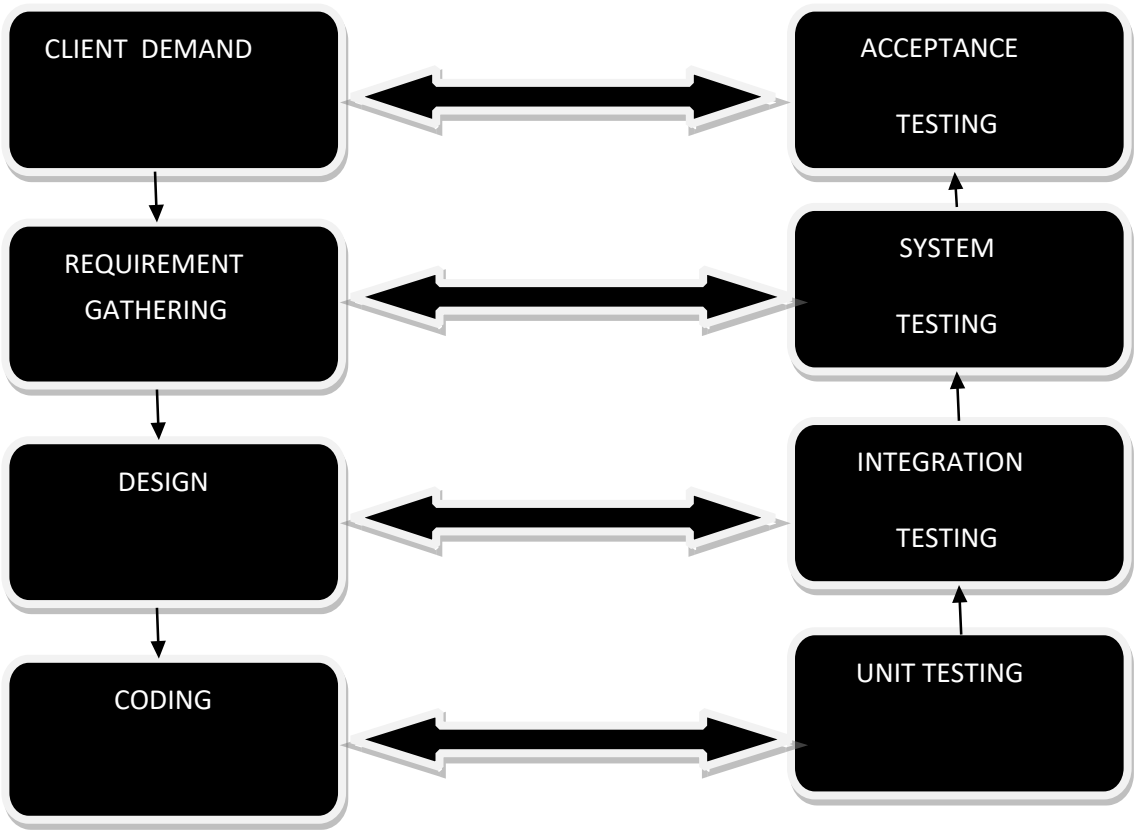
# CHAPTER 6

# TESTING

Testing is the process of evaluating and verifying that a software product or application does what it's supposed to do. The benefits of good testing include preventing bugs and improving performance.

## *Types of Testing:*

- **Unit Testing:** Testing individual components or modules of the system in isolation.
- **Integration Testing:** Testing the interaction between different components or subsystems.
- **System Testing:** Testing the entire system as a whole to ensure it meets the specified requirements.
- **Acceptance Testing:** Testing the system by end-users to verify if it meets their expectations and is ready for deployment
- **Regression Testing:** Retesting previously tested areas of the system after changes have been made to ensure that the changes haven't introduced new defects.



# Testing – Rainwater Harvesting Planner App

Testing the *Rainwater Harvesting Planner* application involves a thorough evaluation to ensure the system functions correctly, provides a good user experience, performs under load, and maintains security and data accuracy.

- ***Functional Testing***

  Each module—such as location selection, weather display, hourly forecast, rain prediction chart, and harvesting suggestions—is tested to verify that it behaves according to the system's requirements. For example, when a user selects a location, the app must correctly fetch and display weather data from the Weather API.

- ***Usability Testing***

  This testing ensures that the user interface is easy to use and visually clear. Users should easily understand how to use dropdowns, read rain predictions, and view suggestions. Feedback is gathered from test users to identify UI/UX flaws and improve layout or responsiveness.

- ***Performance Testing***

  The system is evaluated under high traffic or multiple requests using tools like Postman or JMeter. Load testing ensures that simultaneous users accessing weather data or forecasts do not slow down the app. Stress testing examines the behavior of the system under extreme usage.

- ***Security Testing***

  If the app includes login or stores user preferences, it is essential to ensure that sensitive data like login credentials are securely stored and protected from unauthorized access using secure backend APIs and database validation.

- ***Compatibility Testing***

  The application is tested across various devices (mobile, tablet, desktop) and browsers (Chrome, Firefox, Edge) to ensure consistent layout, functionality, and responsiveness.

- ***Data Integrity Testing***

  Ensures that weather and forecast data fetched from WeatherAPI remains accurate, updated, and consistent across the UI, rain chart, and suggestions. It also verifies correct handling and storage of any data in MongoDB

## ***Testing techniques:***

- **Black Box Testing** - Focuses on testing the software's functionality without any knowledge of the internal code structure. Testers evaluate the system based on inputs and expected outputs. Testing in which the tester doesn't have access to the source code of the software and is conducted

at the software interface without any concern with the internal logical structure of the software known as black-box testing.

- **White Box Testing -** Involves testing the internal workings of an application, such as code structure, logic, and paths. Testing in which the tester is aware of the internal workings of the product, has access to its source code, and is conducted by making sure that all internal operations are performed according to the specifications is known as white box testing

- **Gray Box Testing** - Combines elements of both black box and white box testing, where testers have limited knowledge of the internal code structure. Testing in which the testers should have knowledge of implementation, however, they need not be experts

## ***Principles of Testing***

- Testing shows the presence of defects
- Exhaustive testing is not possible
- Early testing
- Defect clustering
- Pesticide paradox
- Testing is Context-Dependent
- Absence of Errors fallacy

These principles guide testers in designing effective tests, prioritizing efforts, and understanding the limitations of the testing process. By adhering to these principles, software testing can be more focused, efficient, and successful in delivering high-quality software.

# CHAPTER 7

# IMPLEMENTATION

The implementation phase of the Rainwater Harvesting Planner Web App focuses on transforming the planned design and features into a fully functional and user-friendly web application. This stage began with the development of the frontend using HTML, CSS, and JavaScript to create a responsive and visually appealing interface. A dropdown menu was added to allow users to select locations. The interface displays real-time weather data, including temperature, humidity, and rain chances, in an organized and readable format using cards and tables.

The backend development was handled using PHP, which manages the interaction with the WeatherAPI.com service. The application fetches and parses live weather data, including a 7-day forecast and hourly rain probability, based on the user's selected location. This data is dynamically rendered on the page and displayed in both tabular and graphical formats. A live, interactive rain probability chart was implemented using Chart.js, helping users visually interpret the forecast.

To enhance functionality, the app includes dynamic suggestions for rainwater harvesting based on weather conditions. For instance, users receive advice to collect and store rainwater on high-rainfall days and reuse stored water during dry periods. These suggestions are generated based on forecasted data to promote effective water management.

During implementation, careful attention was given to responsive design, ensuring that the application works seamlessly across different screen sizes and devices. The app was thoroughly tested to validate API integration, weather data accuracy, dropdown functionality, chart performance, and the responsiveness of the interface.

The final steps of this phase included organizing the project for deployment, configuring the environment using tools like XAMPP for local testing or hosting it on a web server, and preparing for post-deployment support. This ensures the app remains functional, adaptable, and capable of future updates, such as integrating more advanced analytics or expanding location options. Overall, the implementation phase brought together design, development, integration, and testing to deliver a practical and impactful tool for promoting rainwater harvesting through live weather insights.

.

# FUTURE SCOPE

## FUTUTR SCOPE OF RAINWATER HARVESTING PLANNER APP

The future scope of the Rainwater Harvesting Planner App is promising, especially in the context of increasing water scarcity, rapid urbanization, and the need for sustainable resource management. As climate patterns become more unpredictable, the demand for intelligent tools that help manage and utilize rainwater efficiently will grow. This app can evolve significantly with technological advancements, smart data integration, and user-centric features to make it a valuable tool in promoting water conservation.With integration of real-time satellite weather data, machine learning, and geographic information systems (GIS), the app could provide more precise and localized rainfall predictions. Artificial Intelligence can be employed to generate highly personalized harvesting suggestions, monitor user storage patterns, and offer predictive analysis for water storage based on future rainfall forecasts. Additionally, the app could link with smart sensors or IoT devices installed in rainwater tanks to provide live updates on tank levels and system performance.

### Smart Technology Integration

- **AI-Powered Harvesting Suggestions**: AI can analyze long-term weather trends, tank storage data, and user behavior to generate more accurate and tailored suggestions for storing and using rainwater.
- **GIS-Based Mapping**: Location-specific water runoff and rooftop area mapping can improve estimation of harvestable rainwater for each household or community.

### IoT and Real-Time Monitoring

- **Smart Tank Monitoring**: IoT sensors can be integrated to track rainwater levels in storage tanks and notify users in real time for overflows or shortages.
- **Automated Alerts**: Users could receive SMS or app notifications about expected heavy rainfall, system maintenance, or ideal days to reuse or collect water.

### Wider Community & Government Integration

- **Government Policy Integration**: The app can be linked to local municipal bodies to provide policy recommendations, rebates, and rainwater usage guidelines.
- **Community Sharing Network**: Users can share excess stored water with neighbors or connect to community water banks via the app in water-scarce seasons.

### Enhanced User Experience

- **Mobile App with Offline Mode**: Expanding to a mobile application with offline features ensures broader accessibility in remote or rural areas.
- **Multilingual Support**: Adding support for regional languages will allow users from different linguistic backgrounds to use the app comfortably.

# LIMITATION

While the Rainwater Harvesting Planner App offers a modern solution to promote sustainable water management by predicting rainfall and offering storage suggestions, it is not without its limitations. These limitations can affect its accuracy, efficiency, and overall user experience. One of the key challenges is the app's dependence on real-time weather data and internet connectivity. Any disruption in these services may lead to outdated or missing information, limiting the app's ability to provide timely and accurate recommendations. Furthermore, limitations in scalability, regional adaptability, and integration with physical systems can restrict its broader implementation. Addressing these issues is essential to ensure the app's long-term usability and effectiveness.

- **Dependency on Internet and API Services**: The app requires a stable internet connection to fetch live weather data from third-party APIs. Any connectivity issues or downtime on the weather provider's end can lead to delayed or inaccurate information.
- **Limited Location Support**: The app supports only a fixed set of locations (e.g., near Lucknow). This restricts its usefulness for users in other regions and limits its adaptability on a larger scale.
- **Inaccuracies in Weather Data**: Relying entirely on external weather APIs may lead to occasional inaccuracies in rainfall predictions, which can affect the precision of water harvesting suggestions.
- **No Sensor or IoT Integration**: The app currently lacks integration with real-world sensors or IoT systems, meaning it cannot monitor tank levels, pipe flow, or actual rainfall in real-time.
- **Lack of Infrastructure-Based Customization**: The system does not consider critical factors like roof type, catchment area, and storage capacity, which are essential for estimating accurate water collection volumes.
- **User Accessibility and Learning Curve**: Some users, especially in rural areas or those unfamiliar with technology, may find it difficult to interpret forecast data, charts, or use the app's features efficiently.
- **Security and Data Privacy Concerns**: If user data is collected in future versions, issues related to data protection and privacy may arise, requiring proper safeguards and compliance measures.
- **Limited Offline Functionality**: The app is heavily reliant on live data and cannot operate without internet access, reducing its utility in remote or low-connectivity areas.
- **Scalability Challenges**: As the number of users or supported locations increases, performance issues like longer loading times or data handling limitations may emerge, requiring improved backend infrastructure.

# CONCLUSION

The Rainwater Harvesting Planner App has emerged as a practical and innovative solution for promoting sustainable water management through the use of modern web technologies. By utilizing live weather data, predictive analytics, and interactive visualization, the app aims to empower users to plan, collect, and reuse rainwater efficiently. Throughout its development, the project focused on ensuring usability, responsiveness, and data accuracy. Below are the key areas where the app delivers significant value:

- **Real-Time Weather Forecast Integration:**
  The app uses live data from WeatherAPI to provide accurate and up-to-date weather forecasts, enabling timely planning for rainwater collection.
- **Live Rain Probability Chart:**
  An interactive and dynamic chart visually represents the rain probability trend over the week, making it easier for users to understand and analyze patterns.
- **Localized Data via Dropdown Menu:**
  Instead of a general search, a dropdown list featuring locations near Lucknow allows users to get precise weather data relevant to their area.
- **Smart Harvesting Suggestions:**
  Based on the forecast data, the app provides intelligent suggestions for water harvesting, storage, and reuse, making it actionable and educational for users.
- **User-Friendly Interface:**
  The app is designed with a clean, responsive, and attractive layout, ensuring easy access and navigation for users of all technical backgrounds.
- **Responsive Design for All Devices:**
  With full responsiveness, the app works smoothly across smartphones, tablets, and desktops, ensuring accessibility and performance across platforms.
- **Forecast-Based Decision Making:**
  By linking decisions with accurate forecast data, the app encourages users to engage in proactive water conservation behavior.
- **Future Scalability and Enhancements:**
  The system is built with scalability in mind, allowing future integration with smart devices, IoT-based rainwater tanks, or AI-based notifications.

These features collectively make the Rainwater Harvesting Planner App a forward-thinking initiative that supports eco-friendly practices while leveraging modern technology. It has the potential to create awareness, enhance water resource planning, and contribute to long-term water sustainability efforts.

# REFERENCES

## Reference Books:

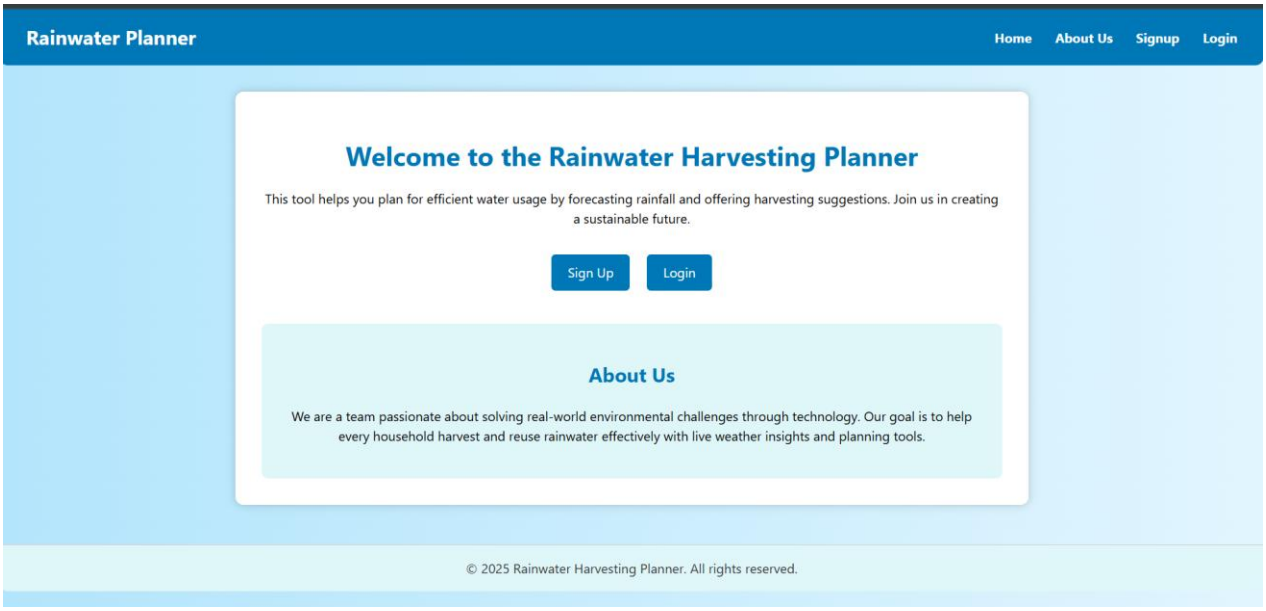

- S. Hoque, *Full-Stack React Projects: Modern Web Development Using React 16, Node, Express, and MongoDB*, 2nd ed. Birmingham, UK: Packt Publishing, 2020.
- V. Subramanian, *Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node*, 2nd ed. New York, NY, USA: Apress, 2019.
- S. Stefanov, *React Up and Running: Building Web Applications*, 1st ed. Sebastopol, CA, USA: O'Reilly Media, 2016.
- L. W. Mays, *Water Resources Engineering*, 2nd ed. Hoboken, NJ, USA: Wiley, 2010.

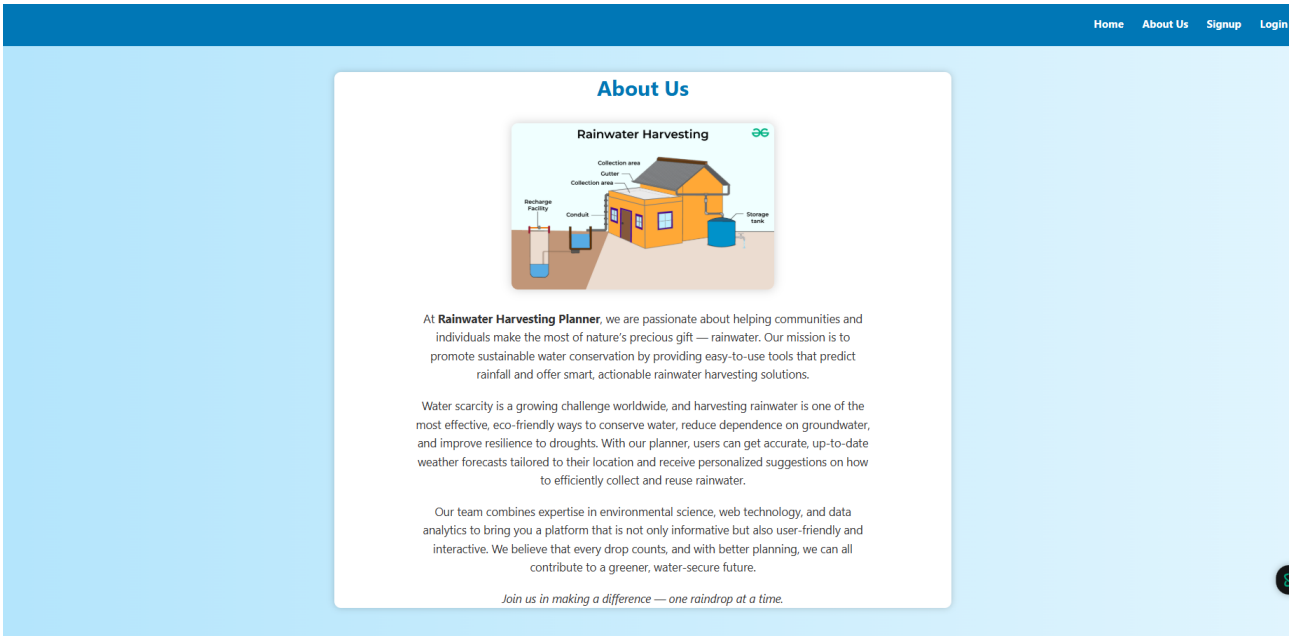

## Reference Websites:

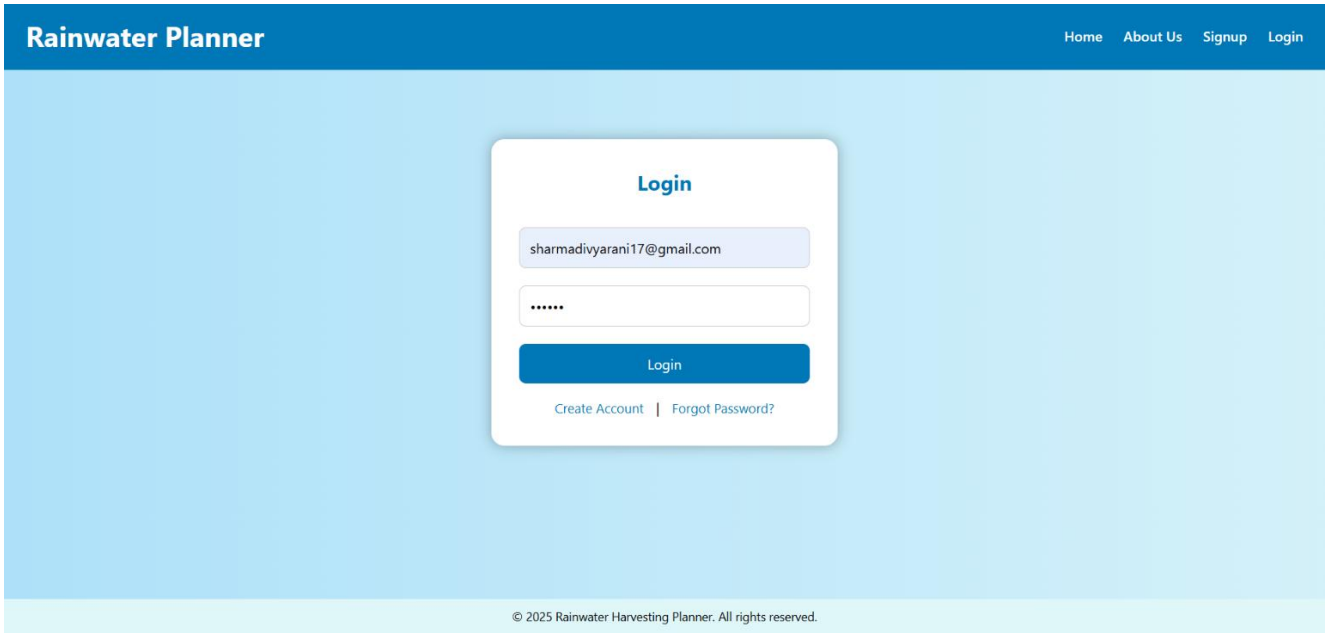

- https://www.freecodecamp.org/news/learn-the-mern-stack-full-tutorial/.
- WeatherAPI.com, "Weather Forecast API," [Online]. Available : https://www.weatherapi.com/.
- Rainwater Harvesting, "Home," [Online]. Available: https://www.rainwaterharvesting.org/
- GeeksforGeeks, "MERN Stack Development Articles, Available: https://www.geeksforgeeks.org/tag/mern-stack/.

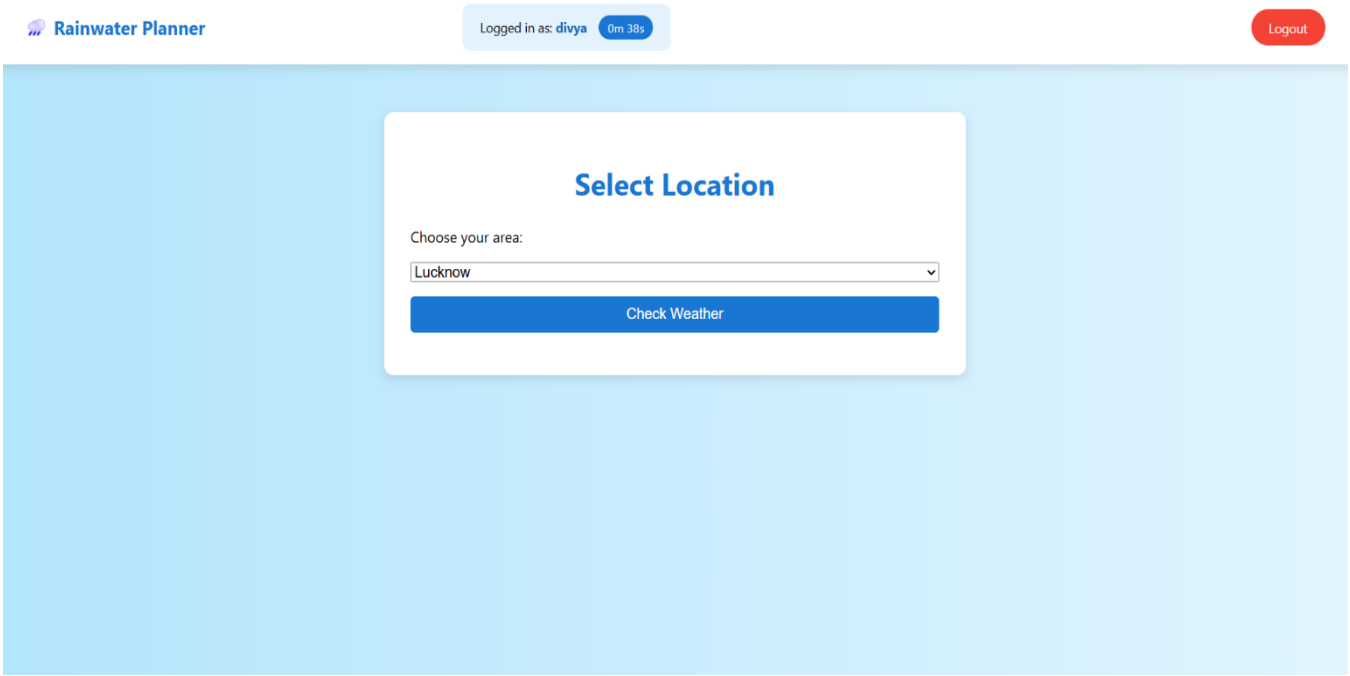
# CHAPTER 5

## SCREENSHOTS OF THE PROJECT




Rainwater Planner
Home
About Us
Signup
Login
Login
sharmadivyarani17@gmail.com
Login
Create Account | Forgot Password?
© 2025 Rainwater Harvesting Planner. All rights reserved.


Rainwater Planner
Logged in as: divya
0m 38s
Logout
Select Location
Choose your area:
Lucknow
Check Weather

# Rainwater Harvesting Planner

Download PDF

## Current Weather in Lucknow

**Temperature:** 27.4°C

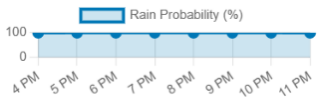
**Humidity:** 100%

**Precipitation:** 2.3 mm

**Condition:** Light rain

### Hourly Forecast (Next 12 Hours)

| Time | Temp (°C) | Rain (%) |
|---|---|---|
| 4 PM | 26.1°C | 100% |
| 5 PM | 26°C | 100% |
| 6 PM | 25.9°C | 100% |
| 7 PM | 27.4°C | 100% |
| 8 PM | 25.9°C | 100% |
| 9 PM | 25.8°C | 100% |
| 10 PM | 25.7°C | 100% |
| 11 PM | 25.7°C | 100% |




## 7-Day Forecast

| Date | Min Temp | Max Temp | Condition | Rain (%) |
|---|---|---|---|---|
| Sun, Aug 3 | 25.7°C | 28°C | Heavy rain | 97% |
| Mon, Aug 4 | 24.5°C | 25.4°C | Heavy rain | 93% |
| Tue, Aug 5 | 24°C | 24.9°C | Moderate rain | 89% |
| Wed, Aug 6 | 24.4°C | 30.3°C | Moderate rain | 83% |
| Thu, Aug 7 | 25.2°C | 32.4°C | Heavy rain | 89% |
| Fri, Aug 8 | 25.5°C | 30.7°C | Moderate rain | 60% |
| Sat, Aug 9 | 25.8°C | 31.4°C | Moderate rain | 87% |

### Suggestions for Rainwater Harvesting

- Heavy rains expected. Install rooftop rainwater collection systems.
- Ensure all storage tanks are clean and covered.
- Use harvested rainwater for irrigation and household cleaning.

← Back to Home

# TABLE


```
{
 "_id": Object Id("64f12ab345a12e7890123456"),
 "username": "divya123",
 "email": "divya@example.com",
 "location": "Lucknow",
 "registered _on": "2025-07-30T10:15:00Z"
}
{
 "_id": Object Id("64f12ac245a12e7890123457"),
 "location": "Lucknow",
 "date": "2025-08-03",
 "forecast": [
  {
   "time": "09:00",
   "temperature": 30.5,
   "rain_ chance": 78
  },
  {
   "time": "12:00",
   "temperature": 32.0,
   "rain_chance": 45
  }
 ]
}
```