Introduction

1.1 Company Profile

InfoWeb Solution

InfoWeb Solution is the leading IT Solutions company based in Surat, India. Since 2012, InfoWeb Solution is a IT Firm which combines beautiful interactive design with intelligent technology such as .NET and REACT, JavaScript frameworks, and PostgreSQL for database management. Here at Speed Limit, InfoWeb Solution understand that having a great website, a print piece, or even a beautiful logo is just not enough. You need results. InfoWeb Solution is a result- focused company. InfoWeb Solution love's tying in creative marketing campaigns to our great work.

Address: - 25, Vatsalya Bungalows, Near S.D Jain Modern School, Vesu, Surat - 395007

Contact No.: - +91 99092 84838

Email: - sunnychevli@gmail.com

1.2 Customer Profile

- The Subscription Management System is an advanced platform designed to simplify subscription management for users and empower businesses to monetize their services through customizable subscription plans. With a sleek, modern interface powered by React Vite and Ant Design, the system offers an intuitive and seamless user experience for all stakeholders.
- Users can register and log in to explore, purchase, and manage subscriptions effortlessly. For businesses, the platform enables them to register, showcase their services, and create subscription plans across three tiers—Normal, Advanced, and Premium—catering to diverse customer needs. Subscribers can monitor their active plans, make changes, or upgrade as required, all within their personalized profiles.
- Built with a robust backend powered by .NET Core Web API 8, Entity Framework, and PostgreSQL, the system ensures secure, reliable, and scalable operations. The real-time capabilities, intuitive workflows, and secure infrastructure make the Subscription Management System an essential tool for users and businesses alike.
- Key features include:
    - Secure Authentication: JWT token-based authentication for user accounts.
    - Core Functions: Login, logout, user registration, password recovery, and subscription management.
    - Business Integration: Business owners can register their enterprises and offer flexible subscription options to customers.
    - User-Friendly Design: A modern, responsive UI that prioritizes accessibility and simplicity for both desktop and mobile users.

1.3 Current System

The **Subscription Management System** caters to four main user roles, each with specific responsibilities and benefits:

1. Admins

Admins play a pivotal role in managing and maintaining the platform's functionality. They are responsible for:

- Overseeing user and business registrations.
- Managing subscription categories and historical data.
- Ensuring smooth operations and system security.

2. Business Owners

Business owners use the platform to monetize their services by offering subscriptions. They can:

- Register their businesses and maintain profiles.
- Create subscription plans in **Normal**, **Advanced**, and **Premium** tiers.
- Monitor user engagement and manage plan performance metrics.

3. Subscribers

Subscribers are end-users who benefit from a user-centric platform. They can:

- Register and log in to explore available subscriptions.
- Purchase and manage their plans with options for upgrades, renewals, or cancellations.
- Access a personalized dashboard to track their active subscriptions and payment history.

4. Technology Stack

The platform is built using cutting-edge technology to deliver an exceptional experience:

- **Frontend**: Developed with React Vite and Ant Design for a fast, modern, and visually appealing user interface.
- **Backend**: Powered by .NET Core Web API 8, providing robust, scalable, and secure functionality.
- **Database**: PostgreSQL ensures reliable and efficient data management.
- **Entity Framework**: Facilitates smooth database interactions and rapid development.

Proposed System

2.1 Scope

The Subscription Management System aims to provide a comprehensive platform for users to manage their subscriptions and for businesses to offer subscription plans effortlessly. It encompasses functionalities for Admins, Business Owners, and Subscribers, ensuring efficient operations and real-time updates. By leveraging .NET Core Web API 8, React Vite with Ant Design, and PostgreSQL, the system delivers a secure, scalable, and user-friendly solution for subscription management.

2.2 Objective

The primary objective of the Subscription Management System is to offer a robust, user-friendly platform for subscription management and business growth. This entails:

- Streamlining subscription workflows for Admins and Business Owners, enabling efficient operations such as business registration, subscription plan creation, and performance tracking.
- Providing subscribers with an intuitive interface to explore, manage, upgrade, or cancel their plans.
- Ensuring secure, real-time operations using JWT authentication and seamless database interactions via PostgreSQL and Entity Framework.
- Enhancing accessibility and scalability through a responsive design powered by React Vite with Ant Design, ensuring consistent functionality across devices.

2.3 Constraints

2.3.1 Hardware Constraints

1. **Server Specifications:**
    - Limited server resources may affect performance and scalability. Sufficient CPU, RAM, and disk space are essential for handling concurrent user requests and real-time updates.
2. **Network Infrastructure:**
    - Bandwidth and latency constraints may impact the responsiveness and reliability of real-time subscription updates.

3. **Storage System:**
    - Adequate storage capacity and performance are required to store and retrieve user data, business profiles, and subscription information efficiently.
4. **Security Hardware:**
    - Constraints in security infrastructure, such as firewalls and encryption tools, may limit the system's ability to safeguard sensitive user data against unauthorized access.
5. **Cost Constraints:**
    - Budget limitations may influence the selection of hardware resources, necessitating cost-effective optimization without compromising performance.

2.3.2 Software Constraints

1. **Database Management System**:
    - PostgreSQL must handle large datasets efficiently while ensuring security and compatibility.
2. **Operating System**:
    - Compatibility with Windows, Linux, and macOS is necessary to support broad accessibility.
3. **Framework and Dependency Limitations**:
    - Version conflicts or limitations in third-party dependencies could hinder flexibility during development and deployment.
4. **Licensing Restrictions**:
    - Compliance with software licensing agreements is essential to avoid legal and operational issues.
5. **Performance Constraints**:
    - Performance bottlenecks in backend APIs or frontend components may affect system scalability.
6. **Security Vulnerabilities**:
    - Proactive measures are required to address potential software vulnerabilities and ensure data confidentiality.

2.4 Advantages

1. **PostgreSQL**:
    - Open-source, cost-effective, and scalable, it offers advanced features such as triggers, stored procedures, and extensive support for modern applications.
2. **React Vite with Ant Design**:
    - Modular and reusable components improve development efficiency.
    - Modern design ensures a professional and seamless user experience.
3. **.NET Core Web API 8**:
    - Cross-platform compatibility enables deployment on various operating systems.
    - Strong integration with Entity Framework simplifies database interactions.
4. **Entity Framework**:
    - Streamlined ORM for database communication reduces development overhead.
5. **Real-Time Functionality**:
    - Integration with robust technologies ensures immediate updates and notifications for users and businesses.

2.5 Limitations

1. **PostgreSQL**:
    - May require experienced database administrators to manage complex queries and scaling.
2. **.NET Core Web API 8**:
    - Limited support for certain third-party libraries compared to other ecosystems like Python or Node.js.
3. **Frontend Complexity**:
    - Achieving consistent cross-platform design may demand additional development and testing efforts.
4. **Resource Constraints**:
    - Development may be impacted by time and budgetary limitations, restricting the implementation of advanced features.

Environment Specification

3.1 Hardware & Software Requirements

- **Hardware Specifications**
  - **Processor: 12th Gen Intel(R) Core(TM) i3-1215U 1.20 GHz (or equivalent)**
  - **RAM: 24 GB (minimum, scalable for higher performance)**
- **Software Specifications**
  - **Operating System: Windows 10, Windows 11**
  - **Frontend Technologies: React, Vite, Ant Design**
  - **Backend Technologies: .NET Core Web API 8**
  - **Database: PostgreSQL with Entity Framework**

3.2 Development Description

The development of the Subscription Management System follows a structured and iterative approach. Leveraging modern tools and technologies, the system is designed to provide a seamless experience for all stakeholders, including Admins, Business Owners, and Subscribers.

I. Requirement Analysis

The first phase involved:

- Identifying core functionalities such as business registration, subscription plan creation, and plan management for users.
- Outlining features tailored to each role, ensuring the platform meets operational and user-experience goals.

II. System Design

The architecture was designed with scalability, security, and efficiency in mind:

- Database Schema: Designed to handle user profiles, businesses, subscription plans (Normal, Advanced, Premium), and transactions.
- Backend Framework: Built using .NET Core Web API 8 to ensure a modular and scalable service-oriented architecture.
- Frontend Framework: Implemented using React Vite with Ant Design for a modern, responsive, and visually appealing user interface.

III. Database Development

- The database was developed using PostgreSQL, optimized for scalability and efficiency.
- Tables were created to manage users, businesses, subscriptions, transactions, and system

configurations.

- Entity Framework was used to simplify database communication and manage relationships between entities.

IV. Backend Development

The backend was developed with the following features:

- API Development: APIs were created for user authentication, subscription management, and real-time updates.
- Authentication: Implemented secure JWT-based authentication for all users.
- Optimization: Temporary in-memory storage was used for processing transient data efficiently, enhancing performance during operations like payment processing.

V. Frontend Development

The frontend development focused on usability and aesthetics:

- Responsive Design: React Vite and Ant Design were used to create a clean, professional user interface optimized for different devices.
- State Management: Redux was utilized to manage application state efficiently, ensuring smooth interactions across components.

VI. Integration and Testing

- Integration Testing: Frontend, backend, and database components were tested for seamless communication and data flow.
- Unit and Functional Testing: APIs and UI components were thoroughly tested to ensure accuracy and reliability.
- End-to-End Testing: Simulated user scenarios to validate the overall system performance.

VII. Deployment Configuration

- The system was deployed on a robust server environment:
    - Operating System: Windows Server or a compatible Linux distribution.
    - Web Server: IIS, Nginx, or Apache for serving the application.
    - Deployment Tools: Docker and CI/CD pipelines were used to streamline deployment and updates.

System Planning

4.1 Feasibility Study

I. Project Scope and Objectives

- Supports four user roles: Admin, Coordinator, Referee, and Viewer, each with tailored functionalities.
- Core objectives include:
    - Real-time score updates.
    - Efficient match and tournament management.
    - Secure user authentication.
    - Intuitive and seamless user experiences.

II. Market Analysis

- Growing demand for real-time sports scoring systems driven by increased digital engagement in sports.
- Potential users include sports organizations, Taekwondo clubs, event coordinators, referees, athletes, and spectators.

III. Technical Feasibility

- Leverages robust technologies:
    - Backend: .NET Core.
    - Frontend: React, Redux.
    - Mobile App: Flutter.
    - Database: SQL Server Management Studio (SSMS).
    - Real-time communication: SignalR.
    - Secure authentication: JWT Tokens and email-based account verification.

IV. Legal and Ethical Feasibility

- Complies with data protection regulations like GDPR and CCPA, ensuring user data privacy and security.
- Promotes fair play, transparency in scoring, and ethical practices.

V. Operational Feasibility

- Seamlessly integrates with existing sports event workflows:
    - Admins manage tournaments and personnel.
    - Coordinators oversee athletes, matches, and scoring.

- Referees perform live scoring.
- Viewers access real-time match updates effortlessly.

4.2 Software Engineering Model

The Agile Model is employed for development, emphasizing iterative progress and adaptability:

- Tasks are divided into smaller iterations lasting 1–4 weeks, minimizing project risk and reducing delivery time.
- Scope, requirements, and iteration plans are defined upfront.
- Regular reviews ensure alignment with goals and enable quick adjustments to evolving requirements.
- Agile principles ensure a collaborative and flexible approach, fostering better project outcomes.

4.3 Risk Analysis

I. Technical Risks

- Potential Issues: System downtime, scalability problems, and technology integration challenges.
- Mitigation:
  - Deploy redundant servers and robust backup systems.
  - Follow well-documented integration practices to ensure seamless interoperability.

II. Project Management Risks

- Potential Issues: Delayed timelines and coordination challenges.
- Mitigation:
  - Employ Agile project management techniques.
  - Conduct regular progress reviews and prepare contingency plans.

III. Operational Risks

- Potential Issues: Inaccurate data entry by referees or coordinators.
- Mitigation:
  - Implement validation checks.
  - Provide user training to improve data entry accuracy.

IV. Security Risks

- Potential Issues: Data breaches and weak authentication.
- Mitigation:
    - Use strong encryption techniques.
    - Implement secure JWT-based authentication.

V. Legal and Compliance Risks

- Potential Issues: Non-compliance with data protection regulations.
- Mitigation:
    - Regularly review and update compliance measures.

System Analysis

5.1 Detailed SRS (Software Requirements Specification)

I. Functional Requirements

Admin Management:

Requirement ID	Requirement Description
FR-ADM-01	Admins can manage subscription plans (add, edit, delete).
FR-ADM-02	Admins can manage users, including viewing, updating, and deactivating accounts.
FR-ADM-03	Admins can view subscription statistics, revenue reports, and analytics.
FR-ADM-04	Admins can set payment options for subscriptions and manage payment gateways.

User Management:

Requirement ID	Requirement Description
FR-USER-01	Users can subscribe to different subscription plans.
FR-USER-02	Users can view and update their subscription details (renew, cancel, change plan).
FR-USER-03	Users can make payments for their subscription via integrated payment gateways.
FR-USER-04	Users can receive notifications about subscription renewals and upcoming payments.

Subscription Management:

Requirement ID	Requirement Description
FR-SUB-01	Users can view available subscription plans and select the desired plan.

Requirement ID	Requirement Description
FR-SUB-02	Admins can manage subscription plan details such as pricing, validity, and features.
FR-SUB-03	The system shall notify users about subscription expiration and renewal.

II. Non-Functional Requirements

Requirement ID	Requirement Description
NFR-01	The system shall ensure high availability, especially during peak usage times.
NFR-02	The system shall maintain data consistency and integrity across all transactions.
NFR-03	The system shall be secure, using data encryption and secure user authentication via JWT tokens.
NFR-04	The system shall provide a responsive, user-friendly interface using React.
NFR-05	The system shall comply with GDPR regulations to ensure user data privacy and protection.

III. Constraints

Requirement ID	Requirement Description
CON-01	The system shall be developed using **.NET Core** for backend APIs.
CON-02	The system shall use **PostgreSQL** for database management.
CON-03	The system shall integrate with third-party payment gateways for payment processing.

5.2 UML Diagrams

5.2.1 Use Case Diagram

- **Admin Use Case:**
- **User Use Case:**

5.2.2 Activity Diagram

- **Admin Activity:**
- **User Activity:**

5.3 Class Diagram

5.4 E-R Diagram

Software Design

6.1 Database Design

CustomerSubscriptions

Field	Type	Constraint
Subscription_Id	integer	Primary Key, Auto-generated
User_Id	integer	Foreign Key (UserAccounts.User_Id)
Plan_Id	integer	Foreign Key (SubscriptionPlans.Plan_Id)
Start_Date	timestamp with time zone	Not Null
End_Date	timestamp with time zone	Not Null
Status	text	Not Null
Payment_Status	text	Not Null
Payment_Method	text	Not Null
Discount_Applied	numeric	Nullable
Created_At	timestamp with time zone	Not Null
Updated_At	timestamp with time zone	Not Null

Feedbacks Table

Field	Type	Constraint
Feedback_Id	integer	Primary Key, Auto-generated
User_Id	integer	Foreign Key (UserAccounts.User_Id)
Vendor_Id	integer	Foreign Key (VendorProfiles.Vendor_Id)
Rating	integer	Not Null
Comments	text	Nullable
Submitted_At	timestamp with time zone	Not Null
Created_At	timestamp with time zone	Not Null
Updated_At	timestamp with time zone	Not Null

Invoices

Field	Type	Constraint
Invoice_Id	integer	Primary Key, Auto-generated
Payment_Id	integer	Foreign Key (Payments.Payment_Id)
Invoice_Number	character varying(50)	Not Null
Issue_Date	timestamp with time zone	Not Null
Due_Date	timestamp with time zone	Not Null
Total_Amount	numeric	Not Null
Created_At	timestamp with time zone	Not Null
Updated_At	timestamp with time zone	Not Null

Notifications

Field	Type	Constraint
Notification_Id	integer	Primary Key, Auto-generated
User_Id	integer	Foreign Key (UserAccounts.User_Id)
Notification_Type	text	Not Null
Message	text	Not Null
Status	text	Not Null
Created_At	timestamp with time zone	Not Null
Sent_At	timestamp with time zone	Nullable
Subject	character varying(255)	Not Null
Is_Email	boolean	Not Null

Payments

Field	Type	Constraint
Payment_Id	integer	Primary Key, Auto-generated
User_Id	integer	Foreign Key (UserAccounts.User_Id)
Plan_Id	integer	Foreign Key (SubscriptionPlans.Plan_Id)
Amount	numeric	Not Null
Payment_Date	timestamp with time zone	Not Null
Payment_Method	text	Not Null
Transaction_Id	text	Not Null
Payment_Status	text	Not Null

Promotions

Field	Type	Constraint
Promotion_Id	integer	Primary Key, Auto-generated
Promotion_Code	character varying(50)	Not Null
Discount_Percentage	numeric	Not Null
Start_Date	timestamp with time zone	Not Null
End_Date	timestamp with time zone	Not Null
Usage_Limit	integer	Not Null
Created_At	timestamp with time zone	Not Null
Updated_At	timestamp with time zone	Not Null
Plan_Id	integer	Foreign Key (SubscriptionPlans.Plan_Id)

SubscriptionPlans

Field	Type	Constraint
Plan_Id	integer	Primary Key, Auto-generated
Vendor_Id	integer	Foreign Key (VendorProfiles.Vendor_Id)
Plan_Name	character varying(255)	Not Null
Description	text	Nullable
Price	numeric	Not Null
Duration_In_Days	integer	Not Null
Features	text	Nullable
Is_Active	boolean	Not Null
Created_At	timestamp with time zone	Not Null
Updated_At	timestamp with time zone	Not Null

6.2 Snapshot for the site

Testing

7.1 Unit Testing

Test Cases for Login

Test Case ID	Test Field	Condition	Expected Result
TC-LOGIN-01	Email	Valid email and correct password	Successful login
TC-LOGIN-02	Email	Valid email but incorrect password	Error message: "Invalid credentials"
TC-LOGIN-03	Email	Invalid email format	Error message: "Invalid email format"
TC-LOGIN-04	Password	Valid email, encrypted password	Successful login
TC-LOGIN-05	Password	Password too short (less than 6 characters)	Error message: "Password too short"

Test Cases for Registration

Test Case ID	Test Field	Condition	Expected Result
TC-REG-01	Email	Valid email format	Successful registration
TC-REG-02	Email	Invalid email format	Error message: "Invalid email format"
TC-REG-03	Password	Password too short (less than 6 characters)	Error message: "Password too short"
TC-REG-04	Password	Encrypted password	Successful registration
TC-REG-05	User Details	All fields valid	New user record created in the database

Test Cases for Forgot Password

Test Case ID	Test Field	Condition	Expected Result
TC-FORG-01	Email	Valid registered email	Password reset email sent
TC-FORG-02	Email	Unregistered email	Error message: "Email not found"
TC-FORG-03	Email	Invalid email format	Error message: "Invalid email format"

Test Cases for User Entries

Test Case ID	Test Field	Condition	Expected Result
TC-ACR-01	Email	Valid email format	Accepts valid email
TC-ACR-02	Password	Valid format	Accepts valid password
TC-ACR-03	Contact Number	Valid number format	Accepts valid contact details
TC-ACR-04	Address	Valid format	Accepts valid address
TC-ACR-05	Role ID	Valid role ID	Accepts valid role

7.2 Integration Testing

Integration Between Backend and Frontend

Test Case ID	Steps	Expected Outcome
INTG-BEFE-01	1. Enter valid credentials in the frontend. 2. Submit login request to backend. 3. Backend authenticates and responds. 4. Frontend updates the UI accordingly.	User successfully logged in
INTG-BEFE-02	1. Fill the registration form in the frontend. 2. Submit registration request. 3. Backend creates a new user record and responds. 4. Frontend confirms successful registration.	User registered successfully
INTG-BEFE-	1. Request a password reset in the frontend.	Password reset email successfully

Test Case ID	Steps	Expected Outcome
03	2. Frontend sends request to the backend. 3. Backend sends reset email instructions. 4. User resets their password.	sent

Future Enhancements

8.1 Personalization and Analytics

- AI-driven recommendations for personalized plans.
- Advanced dashboards with user engagement and churn analytics.

8.2 Flexibility and User Experience

- Pay-as-you-go and multi-plan options.
- Mobile app for managing subscriptions on the go.

8.3 Globalization and Security

- Multi-language and currency support for global reach.
- Two-factor authentication and compliance with security standards.

8.4 Integration and Automation

- Seamless integration with third-party tools like payment gateways and marketing platforms.
- Automated subscription renewals and workflow management.

8.5 Scalability and Performance

- Transition to a scalable cloud infrastructure.
- Microservices architecture for faster feature deployment.

8.6 Advanced Features

- Family and group plans for shared subscriptions.
- Gifting options for subscription sharing.

Glossary

Term	Definition
Admin	A user role responsible for managing subscription plans, user accounts, payments, and overall administrative tasks.
Subscription Plan	A defined package offering access to various services within the system, available for user selection.
Subscriber	A user who has subscribed to a plan and enjoys its benefits based on the active subscription.
Payment Gateway	A service that processes payments for subscription renewals and new sign-ups.
Backend	The server-side component of the Subscription Management System responsible for managing subscription data, payments, and API endpoints, implemented using Dotnet Core.
Frontend	The client-side part of the application that interacts with users, displaying subscription details, managing user inputs, and processing payments. It's implemented using React for web applications and Flutter for mobile.
User Role	A classification within the system that defines the permissions and actions a user can perform, such as Admin, Subscriber, or Guest.
Subscription Renewal	The process by which a subscriber's active plan is extended after its expiration.
Payment	A transaction made by the subscriber to initiate or renew a subscription plan.
Payment Confirmation	A notification or receipt confirming that a subscriber's payment has been successfully processed.
Subscription History	A log that tracks all past subscriptions, renewals, and cancellations of a user.
API	Application Programming Interface; used for communication between the frontend and backend in the system.

Term	Definition
Authentication	The process of verifying a user's identity when accessing the system. Typically done through login credentials or OAuth tokens.
Authorization	The process of granting a user access to specific features based on their role or subscription level.
JWT Token	JSON Web Token; used to securely transfer authentication and authorization details between the client and server.
Subscription Metrics	Analytics and reports on subscriber behavior, including plan choices, usage, and renewal patterns.
Trial Period	A temporary, free access period given to new users to explore subscription benefits before making a purchase decision.
Cancelation	The action by which a user ends their subscription before its natural expiration, often with consequences like data loss or restricted access.
Auto-Renewal	A feature that automatically renews a subscription plan at the end of the billing cycle unless canceled by the user.
Invoice	A document generated to summarize a subscription transaction, including pricing, tax, and payment details.
Email Notification	Automated emails sent to users regarding their subscription status, payment reminders, or promotional offers.
Plan Upgrade/Downgrade	The ability for subscribers to change to a different subscription plan, either with more or fewer benefits.
Customer Support	A service provided to help subscribers with issues related to subscriptions, payments, or technical difficulties.
Subscription Tier	Different levels or categories of subscription plans, each with its own set of benefits and pricing.
Refund	The process of returning money to a subscriber when they cancel their plan within the allowed period or under specific conditions.

Term	Definition
Discount Code	A promotional code that gives users a reduced price on subscription plans or services.
API Rate Limiting	A mechanism used to control the number of requests made to the backend API to prevent system overload.
Subscription Dashboard	A user interface that displays a subscriber's plan details, payment status, and other relevant subscription information.
Renewal Reminder	Automated notifications sent to subscribers to remind them of an upcoming subscription renewal.

Reference

- ✓ **https://www.figma.com/**
- ✓ **https://jwt.io/**
- ✓ **https://medium.com/thesignalgroup**
- ✓ **https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-8.0&WT.mc_id=dotnet-35129-website**
- ✓ **https://www.npmjs.com/**
- ✓ **https://formik.org/docs/overview**