

Solution for

**Building the JARVIS product**To

**LVPEI**

From

**OSI Digital**

Statement of Confidentiality

This document contains proprietary trade secret and confidential information to be used solely for evaluating OSI Digital, Inc., formerly known as OSI Consulting, Inc. (“OSI” “We” or “Us”). The information contained herein is to be considered confidential. Customer by accepting this document, agrees that neither this document nor the information disclosed herein, nor any part thereof, shall be reproduced or transferred to other documents, or used or disclosed to others for any purpose except as specifically authorized in writing by OSI Digital Inc.

**Contents**

[1 About LVPEI 2](#_Toc141294312)

[2 Requirements/Objective of project 3](#_Toc141294313)

[3 Scope 4](#_Toc141294314)

[3.1 Projects/Application Catalogue 7](#_Toc141294315)

[4 Existing Architecture 8](#_Toc141294316)

[4.1 Observations/Cons/Limitations 8](#_Toc141294317)

[5 Challenges 10](#_Toc141294318)

[6 High level solution Architecture 11](#_Toc141294319)

[7 Recommendations 12](#_Toc141294320)

[8 Out of scope 14](#_Toc141294321)

[9 High-level Roadmap 15](#_Toc141294322)

[9.1 Resource Plan 15](#_Toc141294323)

[10 Annexure 16](#_Toc141294324)

[10.1 Application Characteristics 16](#_Toc141294325)

[10.2 HIPAA RULES 17](#_Toc141294326)

[10.3 Subscription & Billing factors to be considered: 17](#_Toc141294327)

[10.4 Azure Cloud based billing products available in marketplace: 19](#_Toc141294328)

[10.5 HL7 Standard format 19](#_Toc141294329)

[10.6 HL7 Supported products/system in India 21](#_Toc141294330)

[10.7 PCI DSS 22](#_Toc141294331)

[10.8 OWASP 23](#_Toc141294332)

# About LVPEI

LVPEI, is a renowned eye care organization based in India, established with the vision of providing high-quality eye care. LVPEI has become a leader in the field of ophthalmology, known for its comprehensive eye care services, research, and education.

LVPEI operates through a network of hospitals, clinics, and vision centers across various locations in India. It offers a wide range of services, including diagnostic evaluations, treatment for various eye conditions, surgeries, and rehabilitation programs. The institute is committed to providing accessible and affordable eye care services to underserved populations through its outreach programs.

In addition to its clinical services, LVPEI is actively involved in research and innovation in the field of ophthalmology. The institute conducts cutting-edge research to develop new treatment techniques, technologies, and solutions for eye diseases and disorders. LVPEI also focuses on training and education, offering various programs for ophthalmologists, optometrists, and allied eye care professionals.

LVPEI's commitment to excellence has earned it recognition and accolades both nationally and internationally. The institute has received several awards for its contributions to eye care and community service. LVPEI continues to strive towards its goal of eliminating avoidable blindness and ensuring comprehensive eye care for all.

# Requirements/Objective of project

LVPEI has successfully developed several innovative projects as part of their research and innovation efforts. Now, LVPEI intends to convert their technology projects into commercial products and release them in the market as Software-as-a-Service (SaaS) offerings that align with cloud-native principles.

LVPEI is going to build the JARVIS product that consists of new apps - One App, Cerebro, Grabi lite, and PupilN.

One App is going to replace the existing HOM application and will provide access to Test applications OM, PupilX, Hawkeye, and One Device. With the One App, users can choose specific tests for specific devices, and the results will be stored safely in databases like EMR, based on what the customer needs.

The JARVIS product mainly works on subscription model. When customers subscribe/buy to a Test platform from the mentioned apps (OM, PUPIL-X, HAWKEYE, or One Device), they can use those specific tests.

The JARVIS product is going to consist of an Analytics portal that helps management/admins to take smart decisions.

# Scope

LVPEI is looking to develop/build **JARVIS** product with the below mentioned modules:

* Product Catalogue Management
  + Cerebro
  + Hawkeye
  + OM
  + PupilX
  + PupilN
  + Grabi Lite
  + One Device)
* Device Management
* Customer/Tenant Management
  + Location configuration
  + Branding
  + Device Registration/configuration
  + Service Integrations & Configurations
* Products Integration with LVPEI Apps
* Integration with Customer EMR
* Managing Customer Reports
  + Search Patient
  + View Case sheet/Report
* User Management
  + Authentication & Authorization
  + Integrating with customer LDAP’s
  + Role Assignments
* Capture Device Performance Statistics
  + Errors capturing
  + Measuring Test times
* Customer Support & Ticketing System
  + Chatbot (limited to only customer staff support)
  + Ticket workflow configurations
  + Ticket Tracking/Dashboards
* Capturing Product Feedback
* User Guide & Trainings
* Analytics Module
  + Product/Device level
  + Patient level
  + Hospital Location Level
  + Sales
  + Tests Statistics
  + Errors/Ticket Monitoring
  + Data & Cloud
  + Inventory
  + Customer Satisfaction/Reviews
* Notifications

Below are various applications that provide functionalities for the above-mentioned modules.

The web application must be responsive web applications to support multiple devices.

The mobile applications are limited to Android & iOS.

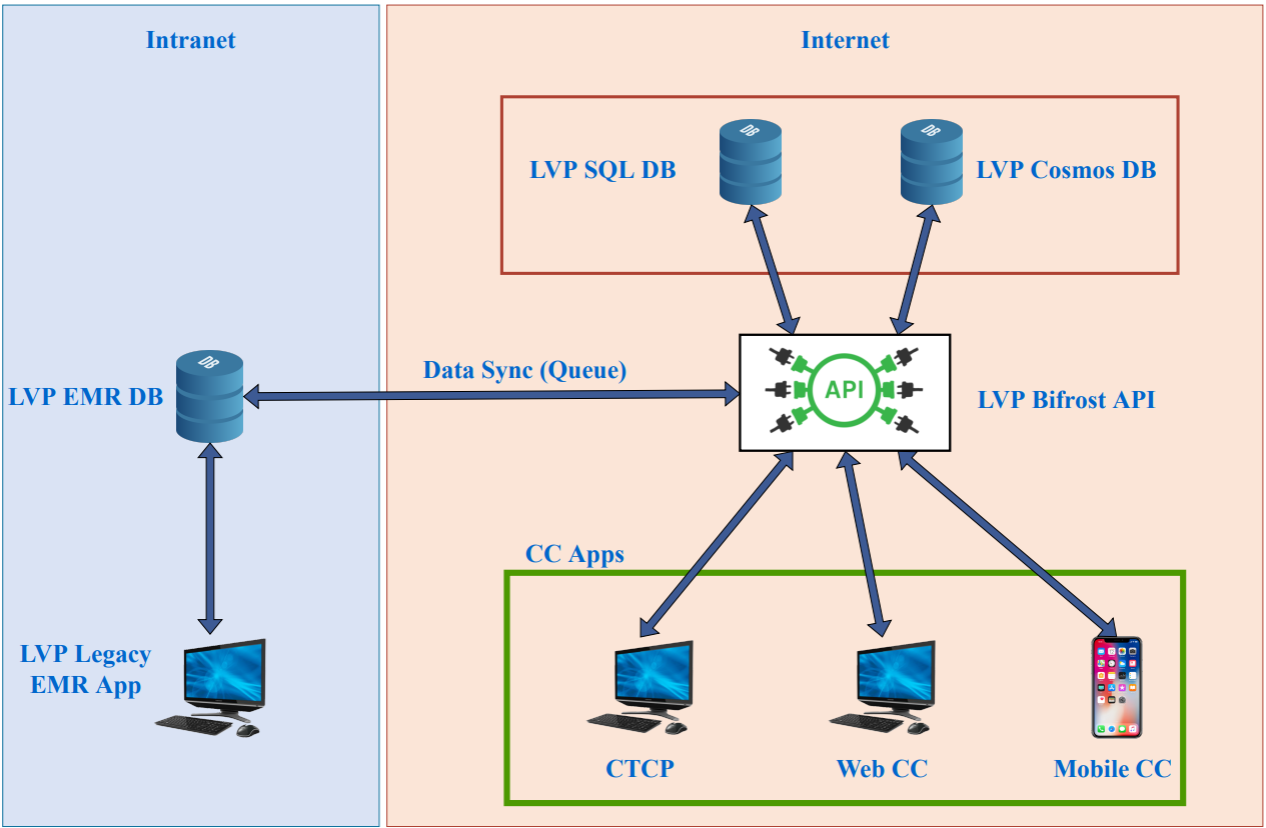
1. **JARVIS** is a **centralized** portal to manage & integrate with the Apps that are subscribed by the Customer. Below are the high-level functionalities for Super Admin, Admin, Examiner & Doctor roles.
   1. Super Admin
      1. Product Catalogue Management
      2. Manage Customers/Tenants
      3. Manage Customer Locations
      4. Manage Customer Branding (limited to Logo)
      5. User & Role Management
      6. Dashboards & Analytics
         1. Product wise sales information
         2. Customer wise subscriptions
         3. Product wise Tickets
         4. Customer Product satisfaction
         5. Top Customers
   2. Admin
      1. Customer User Management (Doctors, Technicians, etc.)
      2. Device Configuration & Management
      3. Test Management
      4. Error/Tickets Tracking
      5. Dashboards & Analytics
         1. Product performance
         * Revenue Graph
         * Test History
         * Recent Orders
         * Tests conducted on each device
         1. Average Test time
         2. Average Billing time
   3. Doctor
      1. Access & View Patient Test Report
   4. Examiner
      1. Conduct patient tests using applications like PupilX, OM, Hawkeye etc.

* **Ticketing System**
  + Define various support levels, Workflows & SLA’s
  + Address the errors/complains raised by customers

## Projects/Application Catalogue

|  |  |  |
| --- | --- | --- |
| Project/  App Name | Type | Description |
| HOM (One App) | Web | HOM (One App) Web HOM is a web application utilized by Examiners to conduct tests for OM, PupilX, Hawkeye, and One Device. For OM and PupilX tests, a virtual device called Pico neo 3 will be used. |
| Android |
| iOS |
| OM | VR Headset (APK) | OM VR Headset(apk) OM is an AI-enabled digital solution that transforms VR headsets into visual field analyzers. The OM module within the One app is responsible for setting up and controlling test sessions, while the OM software on the VR headset conducts the tests. |
| PupilX | VR Headset (APK) | The PupilX test utilizes a swinging flashlight pattern to present light stimuli to a user's eyes. Eye tracking technology captures images of the eyes or measures pupil size through VR headset. The One app's PupilX module sets up and controls test sessions, while the PupilX app on the headset conducts the tests. |
| Hawkeye | Web | Hawkeye is a test that allows examiners to capture patient eye images to diagnose Keratoconus disease. The 3 protocols for Hawkeye test are Anterior Segment Imaging, Tonometry & CT. |
| Android |
| iOS |
| One Device | Web | One Device module is a combination of OM, PupilX, Anterior Segment Imaging, Refraction & Fundus Photography test protocols. |
| Android |
| iOS |
| Cerebro | Android | Cerebro is a mobile application designed for patient self-check-in and appointment management. The application operates through kiosks, allowing patients to conveniently access their upcoming appointment details and make payments. |
| Grabi | Android | Grabi is an accessory that turns smartphones into tools for high-quality eye imaging. The app enables users to capture and share images for clinical review and remote condition management, particularly for post-operative patients and those seeking follow-up care. |
| IOS |
| PupilN |  | PupilN is a standalone version of PupilX that does not require a VR headset or additional control app. It operates on a local Android device, utilizing customized camera attachments connected via Bluetooth and micro-controller USB. PupilN has more testing protocols compared to PupilX like Monocular & Binocular PIPR tests. |
|  |

# Existing Architecture



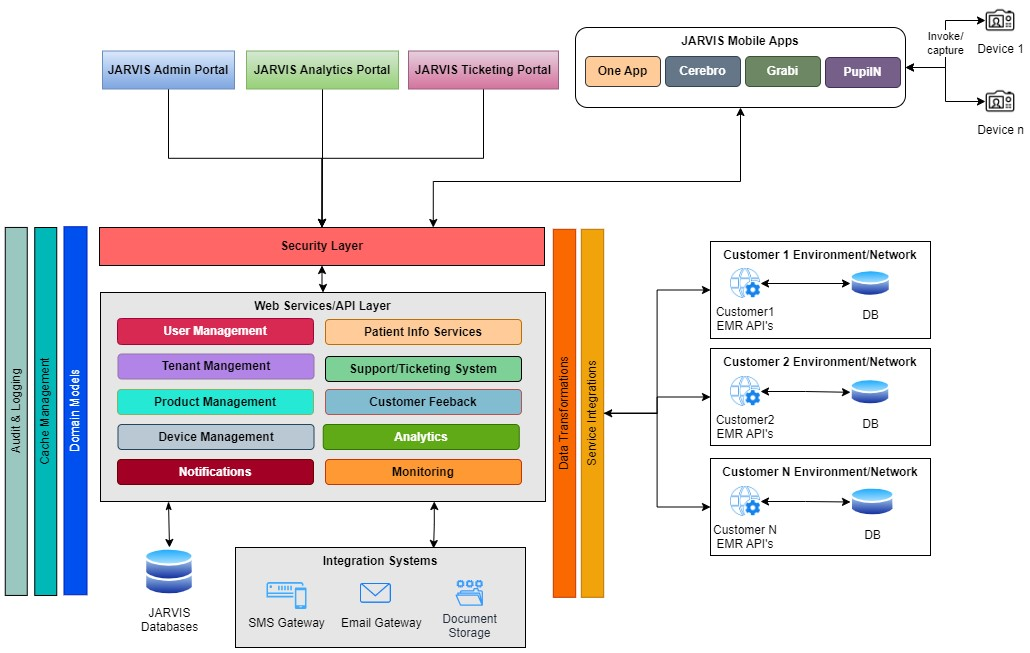
## Observations/Cons/Limitations

* The existing Bifrost system stores patient information and synchronizes patient information from the EMR DB. This is not an effective way as the system is exposing patient information to the outside of EMR system and does not follow source of truth concept.
* Currently the patient data is not stored in standard format like HL7 which makes it harder to integrate with any third party EMR systems.
* Existing system does not have any kind of Super Admin website for managing the Tenants and their configurations
* Not a pure multi-tenant-based application. Requires more generalize the existing code to optimize the multi-Tenant support
* Need to optimize/refactor the existing code to increase the maintainability by implementing loose coupling between the API’s and by improving application/data security
* Needs further optimize the existing applications to improve the application logging/error handling for better monitoring and error tracing
* More number of code projects which will be harder to maintain in the long term. Need to analyze the possibility to combine multiple API projects in terms of business needs
* Require improvement in data security standards to meet application industry standard security norms (like HIPAA)
* Data normalization process needs improvement for better handling of data redundancy
* Currently supporting only one Payment mechanism. Nice to have support for multiple payment mechanisms
* Not having industry standard financial transaction log mechanism for tracing the payment failures
* Not following industry standard deployment processes instead of using manual deployments process

# Challenges

* If the customer already has their own EMR/EHR system and if the customer has patient data privacy policies, customers may raise their concern to replicate the patient information/data in our solution and which is not complaint with standards.
* If the customer has EMR system and to onboard the customer, there may be a significant amount of work involved to integrate with their systems (financial, patient/doctor & appointment system).
* If the customer EMR is down, then maintaining the continuity & syncing of data later to EMR system
* In case the customer wants to deploy and maintain the system within their intranet then we need to identify the mechanism for the deployment process & ongoing support. This kind of deployment may have challenges in terms of support as the support person(s) man need to be at the location physically.

# High level solution Architecture



**Note:** We recommend bring Bifrost and JARVIS under a single portal as we observed there is a significant change in Bifrost as part of integration & data flow.

# Recommendations

* **Business recommendations**
  + We recommend utilizing existing Bifrost system as JARVIS staging platform, and all information will be securely sourced exclusively from the Hospital EMR (Electronic Medical Record) systems which will act as source of truth
  + If the customer **already has EMR system**, then need to derive a mechanism for integration between EMR & JARVIS systems
  + If the customer **does not have any EMR system**, then simply build & deploy the LVPEI specific EMR system & configure the integrations with JARVIS system
  + Define the statistics data need to be captured from the devices
* **Architecture**
* **Security** 
  + **Authentication** - Implement solutions to support different types of authentication mechanisms (Password, LDAP, OAuth etc.). Use multi-factor authentication (MFA) to strengthen authentication mechanism.
  + **Authorization** - Define the Role Matrix during on boarding a new customer
  + **Security & Encryption**
    - Additional to the network level security, consider encrypting the sensitive data while transferring between client application & API.
    - Use industry standard secure protocols like HTTPS, SFTP etc.
    - Consider encrypting the sensitive at all storages level (DB, Cloud etc.)
    - Use Key vaults to store sensitive information
    - Maintain Audit track information who ever access/modify the sensitive data
    - Maintain Security monitoring tools with various alert mechanisms
    - Conduct regular internal audits of the entire system
    - Define robust Backup & Restore mechanism. Regularly back up critical data to a secure location
    - All applications Code must meet industry standards (like OWASP)
* **Implementation Recommendations**
  + Design & Implement piloting with customer
  + Build the SaaS and integration platform by product by Product
  + Build Monitoring & Analytical platform
  + Build E-Commerce platform & Billing platform
* **Branding**
  + Maintain a separate Mobile application for each Tenant to have flexibility with their branding, registration, backend integration & other custom configurations. On the other side there may be a little bit of additional work required to build & deploy multiple applications to the mobile app stores.
  + Perform Web application branding at the domain/sub domain level for better user experience. Example: aims.jarvis.com
* **Compliance** 
  + The Applications must in line with HIPAA standard compliances (See **Annexure: HIPAA RULES**)
  + Data exchange between the systems should be in line with HL7 Standards (See **Annexure: HL7 Standard format**)
  + OWASP security standards (See **Annexure: OWASP**)
  + Payment gateway standards – PCI DSS (See **Annexure: PCI DSS**)
* **Subscription Models**
  + Define different types of Subscription models and Pricing (Fixed, Variable, metered, Free Trail etc.). You can consider various factors mentioned below:
    - User Seats/Concurrent Users
    - Data Storage
    - API usage (limit to certain number of calls)
    - Billing frequency
    - Overage Charges (Charges when customer exceeds their allocated resources)
* **Deployment & Customer Onboarding**
  + Automated deployments for Web Applications
  + Customer Onboarding process (if required integrations):
    - Sign the contract agreement
    - Customer Environment Discovery
    - Define required API’s and collaborate with customer’s IT
    - Develop/Deploy integration on UAT environment
    - Customer UAT Signoff
    - Production Deployment/Configuration

# Out of scope

* Any requirements not mentioned in the scope section
* Patient appointment Registration/Creation
* Patient/Doctor Mobile Applications (CCP & CCD) integrations
* Any kind of infrastructure setup
* Hardware procurement & setup
* Application performance testing
* Any kind of data imports/exports
* Customizable dashboards

# High-level Roadmap

|  |  |
| --- | --- |
|  | Deliverables & Activities |
| Month 1 & 2 | * Requirement gathering * Sprints & Stories creation * Product Architecture & Design * Develop new Base platform/framework for Authentication/Authorization, Mutli-tenancy, Device Management |
|
|
| Month 3 | * EMR DB integration * Complete integration of one selected App |
| Month 4 | * Integrations for the remaining Apps * Ticketing system |
| Month 5 | * Build Analytical framework * Build & Integrate required Analytical dashboards |
| Month 6 | * Cerebro App development & Integration * Manage Patient Reports |

## Resource Plan

|  |  |  |
| --- | --- | --- |
| Role | Resource Type | Resource Count |
| Architect | Partial | 1 |
| Backend Developer | Full Time | 2 |
| Frontend/Mobile Developer | Full Time | 2 |
| QA | Full Time | 1 |
| Cloud Infra | Partial | 1 |

# Annexure

## Application Characteristics

Based on the provided characteristics, the description outlines the key features and requirements for the SaaS (Software as a Service) product. Here's a summary of the characteristics:

1. SaaS Product: The software is offered as a service, accessible through the internet, and does not require installation on the client's end.
2. Multitenant: The SaaS application is designed to serve multiple customers (tenants) from a single shared instance, ensuring efficiency and scalability.
3. Open for Integrations & Data Transformation: The SaaS product allows integration with other third-party applications and data sources, enabling seamless data exchange and transformation.
4. API (Application Programming Interface): The SaaS product provides APIs that allow external applications to interact and integrate with its functionality.
5. Billing and Metering: The SaaS product includes capabilities for billing customers based on usage and metering resources as needed.
6. Configurable: The SaaS application offers various configuration options to adapt to different customer requirements and preferences.
7. Compliance with HIPAA: The SaaS product meets the security and privacy standards required for handling sensitive healthcare information as per the Health Insurance Portability and Accountability Act (HIPAA).
8. Secure and Integrated with Client AD's Storage: The SaaS application is designed with security in mind and allows integration with the client's Active Directory for authentication and authorization purposes.
9. Leverage Branding: The SaaS product offers branding options, allowing customers to customize the appearance and user experience to match their brand identity.
10. Cloud-Native: The SaaS product is built using cloud-native technologies and takes advantage of the scalability and flexibility offered by cloud platforms.
11. Responsive and Mobile Integration: The SaaS application is designed to be responsive, providing an optimal user experience across different devices, including mobile devices.
12. No Customer Specific Customization: The SaaS product does not allow individual customizations for each customer to maintain uniformity and reduce complexity.
13. Security - Two-Factor Authentication: The SaaS application supports two-factor authentication, adding an extra layer of security to user logins.
14. API Versioning Support: The SaaS product's APIs are designed to support versioning, ensuring backward compatibility as the product evolves.
15. 90% Availability: The SaaS product aims to achieve 90% uptime, providing a highly available service to customers.

## HIPAA RULES

1. Access Controls: Implementing mechanisms to limit access to ePHI to only authorized individuals who need it to carry out their job responsibilities. This can include user authentication, role-based access controls, and activity monitoring.
2. Encryption and Decryption: Encrypting ePHI stored in databases to protect it from unauthorized access or disclosure.
3. Audit Controls: Implementing hardware, software, and/or procedural mechanisms to record and examine activity in information systems that contain or use ePHI.
4. Integrity Controls: Implementing measures to ensure that ePHI is not altered or destroyed in an unauthorized manner. This can include data backup and recovery processes.
5. Transmission Security: Implementing security measures to protect ePHI during transmission over electronic networks, such as the internet or private networks.
6. Physical Safeguards: Implementing physical measures to protect the physical infrastructure that houses databases containing ePHI, such as access controls to data centers and server rooms.

At present, all applications retrieve data from the Bifrost system, which is stored in the Bifrost database. However, we are uncertain about the hospitals' willingness to share patient data and case sheet information with the Bifrost system directly. Our proposal is to utilize Bifrost as a staging platform, and all information will be sourced exclusively from the Hospital EMR (Electronic Medical Record) systems. This approach will be followed as long as hospitals and clients maintain their individual EMR systems. Ensuring compliance with this essential requirement is a crucial aspect of determining the optimal solution.

## Subscription & Billing factors to be considered:

When building a Software-as-a-Service (SaaS) product, several parameters need to be considered for billing, especially concerning the infrastructure. The billing model you choose will depend on your product's features, target audience, and overall business strategy. Here are some key parameters to consider for billing as part of the infrastructure:

1. **Resource Usage**: Monitor and track the usage of resources, such as computing power (CPU), memory, storage, and bandwidth. Depending on the intensity of usage, you can implement different pricing tiers or charge customers based on their resource consumption.
2. **Pricing Tiers and Plans**: Offer different pricing tiers or plans to cater to various customer segments with varying needs and budgets. Common tiers include basic, pro, and enterprise levels, each with different features and resource allocations.
3. **Metered Billing**: Implement a metered billing model where customers pay for what they use. This approach is suitable for services where usage can fluctuate, and customers prefer to pay based on actual usage.
4. **Fixed Fee vs. Variable Fee**: Decide whether to charge a fixed fee for a predefined set of features/resources or a variable fee based on actual usage. A combination of both approaches is also possible.
5. **Free Trial and Freemium**: Consider offering a free trial or freemium version with limited features to attract potential customers and allow them to explore the product before committing to a paid plan.
6. **Data Storage**: If your SaaS product involves data storage, determine how much storage space is allocated to each customer, and decide if additional storage should be charged separately.
7. **API Usage**: If your SaaS product provides APIs for integration with other services, consider whether to charge based on the number of API calls or provide a certain number of free API calls with the option to purchase additional ones.
8. **Geographic Regions**: If your infrastructure is spread across multiple regions, pricing may differ based on data center locations and accessibility.
9. **User Seats or Concurrent Users**: Consider charging based on the number of user seats (individual user accounts) or concurrent users accessing the platform simultaneously.
10. **Support Levels**: Offer different support levels (e.g., basic, standard, premium) with varying response times and assistance options, and price them accordingly.
11. **Service Level Agreements (SLAs)**: SLAs may include uptime guarantees, response times, and performance benchmarks. You can use SLAs to differentiate pricing for customers requiring higher reliability and performance.
12. **Overage Charges**: Determine how to handle overage charges when customers exceed their allocated resources. You can either charge extra for additional usage or automatically upgrade the customer to the next pricing tier.
13. **Billing Frequency**: Decide on the billing frequency, whether it's monthly, annually, or another interval that suits your customers' preferences.
14. **Payment Methods**: Offer various payment options, such as credit cards, bank transfers, and digital wallets, to cater to customers from different regions.
15. **Promotions and Discounts**: Consider offering promotions, discounts, or referral incentives to attract new customers and retain existing ones.

The pricing and billing models can significantly impact on your product's success and customer satisfaction. It's essential to keep an eye on market trends, competitor offerings, and customer feedback to continually optimize your billing strategy. Additionally, consider seeking legal and financial advice to ensure compliance and transparency in your billing practices.

## Azure Cloud based billing products available in marketplace:

* **Stripe Billing**: Stripe provides an integration with Azure through its REST APIs, allowing SaaS businesses to handle payment processing, billing, and subscriptions in their applications running on Azure.
* **Chargebee**: Chargebee has an integration with Azure that allows businesses to manage subscriptions, billing, and revenue operations for SaaS products hosted on the Azure cloud.
* **Recurly**: Recurly offers integration with Azure, enabling SaaS businesses to manage subscription billing, revenue optimization, and analytics within their Azure-hosted applications.

## HL7 Standard format

Below is a simple example of an HL7 version 2.x message for patient registration:

MSH|^~\&|SendingApp|SendingFac|ReceivingApp|ReceivingFac|20230721120000||ADT^A04|MSG123456789|P|2.3.1

EVN|A04|20230721120000

PID|1|123456789|987654321||Doe^John||19900101|M|||123 Main St^^City^State^ZIP||+1 555-123-4567|||S|||1234567890

PV1|1|I|ER^Room1^Bed1|||^^^Dr. Smith|||E|||12345^Smith^John|Emergency|||||||||12345^Smith^John||||||||||||||||||||||||||20230721120000

Break down explanation for the segments in the above example:

1. **MSH (Message Header)**:

* Field 1: Field Separator (^)
* Field 2: Encoding Characters (~&)
* Field 3: Sending Application (SendingApp)
* Field 4: Sending Facility (SendingFac)
* Field 5: Receiving Application (ReceivingApp)
* Field 6: Receiving Facility (ReceivingFac)
* Field 7: Date/Time of Message (20230721120000 - July 21, 2023, 12:00:00)
* Field 8: Security (blank in this example)
* Field 9: Message Type (ADT^A04 - Patient Registration)
* Field 10: Message Control ID (MSG123456789)
* Field 11: Processing ID (P)
* Field 12: Version ID (2.3.1)

1. **EVN (Event Type)**:
   * Field 1: Event Type Code (A04 - Patient Registration/Update)
2. **PID (Patient Identification)**:
   * Field 1: Set ID - PID (1)
   * Field 2: Patient ID (123456789)
   * Field 3: Patient Identifier List (987654321)
   * Field 4: Alternate Patient ID (blank in this example)
   * Field 5: Patient Name (Doe^John)
   * Field 6: Date of Birth (19900101 - January 1, 1990)
   * Field 7: Patient Sex (M - Male)
   * Field 8: Patient Alias (blank in this example)
   * Field 9: Patient Race (blank in this example)
   * Field 10: Patient Address (123 Main St^^City^State^ZIP)
   * Field 11: Phone Number ( +1 555-123-4567)
   * Field 12: Marital Status (blank in this example)
   * Field 13: Patient Religion (blank in this example)
   * Field 14: Patient Account Number (blank in this example)
3. **PV1 (Patient Visit)**:
   * Field 1: Set ID - PV1 (1)
   * Field 2: Patient Class (I - Inpatient)
   * Field 3: Assigned Patient Location (ER^Room1^Bed1)
   * Field 4: Admission Type (blank in this example)
   * Field 5: Preadmit Number (blank in this example)
   * Field 6: Prior Patient Location (blank in this example)
   * Field 7: Attending Doctor (^^^Dr. Smith)
   * Field 8: Referring Doctor (blank in this example)
   * Field 9: Consulting Doctor (blank in this example)
   * Field 10: Hospital Service (E - Emergency)
   * Field 11: Temporary Location (blank in this example)
   * Field 12: Preadmit Test Indicator (blank in this example)
   * Field 13: Re-admission Indicator (blank in this example)
   * Field 14: Admit Source (Emergency)
   * Field 15: Ambulatory Status (blank in this example)
   * Field 16: VIP Indicator (blank in this example)
   * Field 17: Admitting Doctor (12345^Smith^John)
   * Field 18: Patient Type (Emergency)
   * Field 19: Visit Number (blank in this example)
   * Field 20: Financial Class (blank in this example)
   * Field 21: Charge Price Indicator (blank in this example)
   * Field 22: Courtesy Code (blank in this example)
   * Field 23: Credit Rating (blank in this example)
   * Field 24: Contract Code (blank in this example)
   * Field 25: Contract Effective Date (blank in this example)
   * Field 26: Contract Amount (blank in this example)
   * Field 27: Contract Period (blank in this example)
   * Field 28: Interest Code (blank in this example)
   * Field 29: Transfer to Bad Debt Code (blank in this example)
   * Field 30: Transfer to Bad Debt Date (blank in this example)
   * Field 31: Bad Debt Agency Code (blank in this example)
   * Field 32: Bad Debt Transfer Amount (blank in this example)
   * Field 33: Bad Debt Recovery Amount (blank in this example)
   * Field 34: Delete Account Indicator (blank in this example)
   * Field 35: Delete Account Date (blank in this example)
   * Field 36: Discharge Disposition (blank in this example)
   * Field 37: Discharged to Location (blank in this example)
   * Field 38: Diet Type (blank in this example)
   * Field 39: Servicing Facility (blank in this example)
   * Field 40: Bed Status (blank in this example)
   * Field 41: Account Status (blank in this example)
   * Field 42: Pending Location (blank in this example)
   * Field 43: Prior Temporary Location (blank in this example)
   * Field 44: Admit Date/Time (blank in this example)
   * Field 45: Discharge Date/Time (blank in this example)
   * Field 46: Current Patient Balance (blank in this example)
   * Field 47: Total Charges (blank in this example)
   * Field 48: Total Adjustments (blank in this example)
   * Field 49: Total Payments (blank in this example)
   * Field 50: Alternate Visit ID (blank in this example)
   * Field 51: Visit Indicator (blank in this example)
   * Field 52: Other Healthcare Provider (blank in this example)
   * Field 53: Service Episode Description (blank in this example)
   * Field 54: Service Episode Identifier (blank in this example)
   * Field 55: Service Episode Identifier (blank in this example)
   * Field 56: Service Episode Identifier (blank in this example)
   * Field 57: Service Episode Identifier (blank in this example)
   * Field 58: Service Episode Identifier (blank in this example)
   * Field 59: Service Episode Identifier (blank in this example)
   * Field 60: Service Episode

## HL7 Supported products/system in India

* **Practo**: Practo is a well-known healthcare platform in India that provides EMR solutions and supports HL7 integration for interoperability.
* **CureMD**: CureMD is a cloud based EMR system that offers HL7 support and aims to streamline healthcare operations.
* **Insta Health Solutions**: Insta Health Solutions is an EMR system used in various healthcare facilities in India, and it supports HL7 standards for data exchange.
* **Mediware**: Mediware is an EMR system designed for hospitals and clinics in India, and it includes HL7 capabilities for seamless data sharing.
* **Lybrate**: Lybrate is an online healthcare platform that provides an EMR system and supports HL7 integration for efficient data exchange.
* **HealthcareMagic**: HealthcareMagic offers an EMR system with HL7 support to enhance communication and data sharing between healthcare providers.
* **NABH-IMS**: The National Accreditation Board for Hospitals & Healthcare Providers (NABH) has an Integrated Management System (IMS) that includes EMR functionality and supports HL7 for data exchange.
* **MocDoc**: MocDoc is an EMR, and practice management solution used in healthcare facilities in India, and it offers HL7 integration for interoperability.
* **Sensibull**: Sensibull is an EMR system with HL7 support, designed to cater to the needs of small and medium-sized healthcare facilities.
* **Medinous**: Medinous is an EMR system designed for hospitals and healthcare providers, and it includes HL7 capabilities for data exchange with other systems.

## PCI DSS

CI DSS stands for Payment Card Industry Data Security Standard. It is a set of security standards designed to protect cardholder data and ensure the secure handling of credit card information during payment transactions. PCI DSS is maintained by the Payment Card Industry Security Standards Council (PCI SSC), which is a collaborative organization founded by major card brands like Visa, MasterCard, American Express, Discover, and JCB.

The PCI DSS consists of a comprehensive set of requirements that apply to any organization that handles, processes, or stores payment card data. These requirements are grouped into six main categories:

**1. Build and Maintain a Secure Network and Systems:**

* Install and maintain a firewall configuration to protect cardholder data.
* Do not use vendor-supplied default passwords and security parameters.
* Protect cardholder data through encryption during transmission over open, public networks.

**2. Protect Cardholder Data:**

* Protect stored cardholder data using encryption and secure cryptographic mechanisms.
* Mask cardholder data when displayed, except for necessary business purposes.
* Restrict access to cardholder data on a need-to-know basis.

**3. Maintain a Vulnerability Management Program:**

* Use and regularly update anti-virus software or programs.
* Develop and maintain secure systems and applications by applying patches and security updates.

**4. Implement Strong Access Control Measures:**

* Restrict access to cardholder data by business need-to-know.
* Assign a unique ID to each person with computer access and implement authentication mechanisms.
* Restrict physical access to cardholder data.

**5. Regularly Monitor and Test Networks:**

* Track and monitor access to network resources and cardholder data.
* Regularly test security systems and processes.
* Maintain an information security policy and review it annually.

**6. Maintain an Information Security Policy:**

* Establish and maintain a company-wide information security policy that addresses the protection of cardholder data.
* Provide clear guidance on security responsibilities and expectations for all employees.

## OWASP

OWASP (Open Web Application Security Project) is a well-known nonprofit organization focused on improving the security of software and web applications. They provide a list of the top web application security risks, known as the OWASP Top 10. These are considered the most critical and prevalent vulnerabilities that developers should be aware of and take measures to address. The OWASP Top 10 is regularly updated to reflect the evolving threat landscape. As of my last knowledge update in September 2021, the most recent version is OWASP Top 10 2021. Here are the common vulnerability categories listed in the OWASP Top 10 2021:

* **Injection:** This vulnerability occurs when untrusted data is sent to an interpreter as part of a command or query, leading to malicious code execution. Common examples include SQL injection and OS command injection.
* **Broken Authentication:** This vulnerability refers to flaws in authentication and session management mechanisms, allowing attackers to compromise user accounts, passwords, keys, or session tokens.
* **Sensitive Data Exposure:** It involves the exposure of sensitive data (such as credit card numbers, personal information, or passwords) due to weak encryption or improper handling.
* **XML External Entities (XXE):** XXE vulnerabilities occur when an application processes XML input that contains external entity references, leading to information disclosure or server-side request forgery (SSRF) attacks.
* **Broken Access Control:** This vulnerability allows attackers to bypass access controls and gain unauthorized access to functionality or data.
* **Security Misconfiguration:** It results from insecure default configurations, incomplete or ad hoc configurations, and unnecessary features that can be exploited by attackers.
* **Cross-Site Scripting (XSS):** XSS occurs when an attacker injects malicious scripts into web pages viewed by other users, potentially leading to the theft of sensitive information or hijacking of user sessions.
* **Insecure Deserialization:** Deserialization vulnerabilities occur when untrusted data is used to exploit flaws in the deserialization process, leading to remote code execution, DoS attacks, or data tampering.
* **Server-Side Request Forgery (SSRF):** SSRF vulnerabilities allow attackers to make requests from the server to other internal or external systems, potentially leading to data exposure or attacks against internal systems.
* **Insecure Software Components:** This refers to the use of vulnerable or outdated software components (libraries, frameworks, etc.) that can be exploited by attackers.