AFOURATHON 2023

HACKATHON

Product Name	**Student and Library Base Management System**
Prepared by	**Harsh Bihani** **Prajwal Randive**

Hello!

Are you an over worked *Class Representative* of your department of engineering running behind teachers for managing students.

Or

Do you work in the *administration department* of your institution & are frustrated because of ill-managed data which constantly hinders your efficiency.

Or

Are you a *student* who wants to register for a book from the library but can't seem to do it efficiently without running behind the librarian .

Or

Are you the tired *librarian* itself who cant seem to keep track of the books in order, due to the students being careless.

Well, look no further because we have created a website, specially curated for you in order to correct

these slight inconveniences and improve your productivity in your respective institution.

Let me present to you the

STUDENT AND LIBRARY BASE MANAGEMENT SYSTEM

OR MORE EASILY KNOWN AS

SEL-BMS

As per the problems discussed above, lets see how our product aims to solve them and delve deep into its working.

Solution

1. ***Solution Purpose***

   SEL-BMS aims to solve many persnickety issues of any institution with its user-friendly website specially curated to target the issues specified above and more. Its mainly aimed to solve the problems faced by the college and department administrator, the college librarian and the students.

   It would help the students as such:

   - Students can keep track of their data available to the institution and also update it whenever required, which would help them during intra-college applications and scholarship details.
   - Students would be able to see the available books in the College library and also see the books which are in use by other students so they can schedule their studies likewise.
   - Students will be able to request books directly from the library administrator via the website without any hassle. They will also be able to view the scheduled return dates for books that are currently in use.
   - It would also help the Class representative (CR) to keep a better track of students and books, and also have better access to the details of professors and administrators for the smooth functioning of their respective class.

   It would help the admins as such:

   - The admins of college constantly face a shortage and irregularity of data of students due to the ill-maintained records. This problem would be eliminated by our site as it would constantly update the database.

- Admins would have a greater access to the details which would be segregated and marked, such that anyone with the authority could use it for their work.
- Admins would also have the authority to add, update or delete a student's profile in the light of any issue.
- It would also help the college authorities to create certificates and hall-tickets with ease and cross-check the details if necessary.
- Lastly, the better availability of student data would help the college to make new reforms, implement policies and overall benefit the college.

It would help the librarians as such:

- The librarians would have a crystal-clear database of all the books present inside the library via proper documentation entered on the website.
- This would solve the need of maintaining physical database which most of the times is neglected causing the efficiency of the library to drop down making it hard for the students to borrow books and for the librarians to lend the same.
- They would be able to add, update or delete any book's details according to their need.
- They would be able to keep track of all the books which are available for the students and also the books which are currently in use (lent) by other students.
- Librarian would have a better track record so as to prevent loss of any book and issue the penalties likewise.
- They would be able to see the scheduled return dates of the books and thus, impose penalties in case of late returns.
- Lastly, they would be able to accept or deny the request of any student for a certain book on the website itself.

2. Scope and No Scope

SCOPE:

1. User registration and login functionality for students, administrators, and librarians.

2. Student profile management.

3. Access to a comprehensive database of available books in the college library.

4. Visibility of books currently in use by other students, enabling students to plan their studies accordingly.

5. Book request feature for students to directly request books from the library administrator.

6. Display of scheduled return dates for books currently in use.

7. Enhanced capabilities for class representatives (CR) to track students, books, and access details of professors and administrators.

8. Regular updating of student data to ensure accurate and well-maintained records for administrators.

9. Authority for administrators to add, update, or delete student profiles as needed.

10. Improved availability of student data to facilitate reforms, policy implementation, and overall college benefits.

11. A well-maintained database of all books in the college library thus, eliminating the need for physical database maintenance, improving library efficiency for both students and librarians.

12. Book management capabilities for librarians, including adding, updating, and deleting book details.

13. Tracking of available books and those currently lent to students.

14. Efficient record-keeping to prevent book theft and impose penalties for late returns.

15. Acceptance or denial of book requests by students through the website.

NO SCOPE:

1. Online examination module and automatic generation of documents via the available details.

2. Integration with third-party email marketing services. (unless explicitly mentioned).

3. Development of a mobile application.

4. Inventory management functionality beyond book tracking.

5. Integration with specific student information systems used by the Institutions (unless explicitly mentioned).

6. Social media integration for user authentication beyond traditional registration and login.

7. Payment gateway integration for fee collection (unless explicitly mentioned).

8. Integration with external systems beyond the mentioned use cases for students, administrators, and librarians.

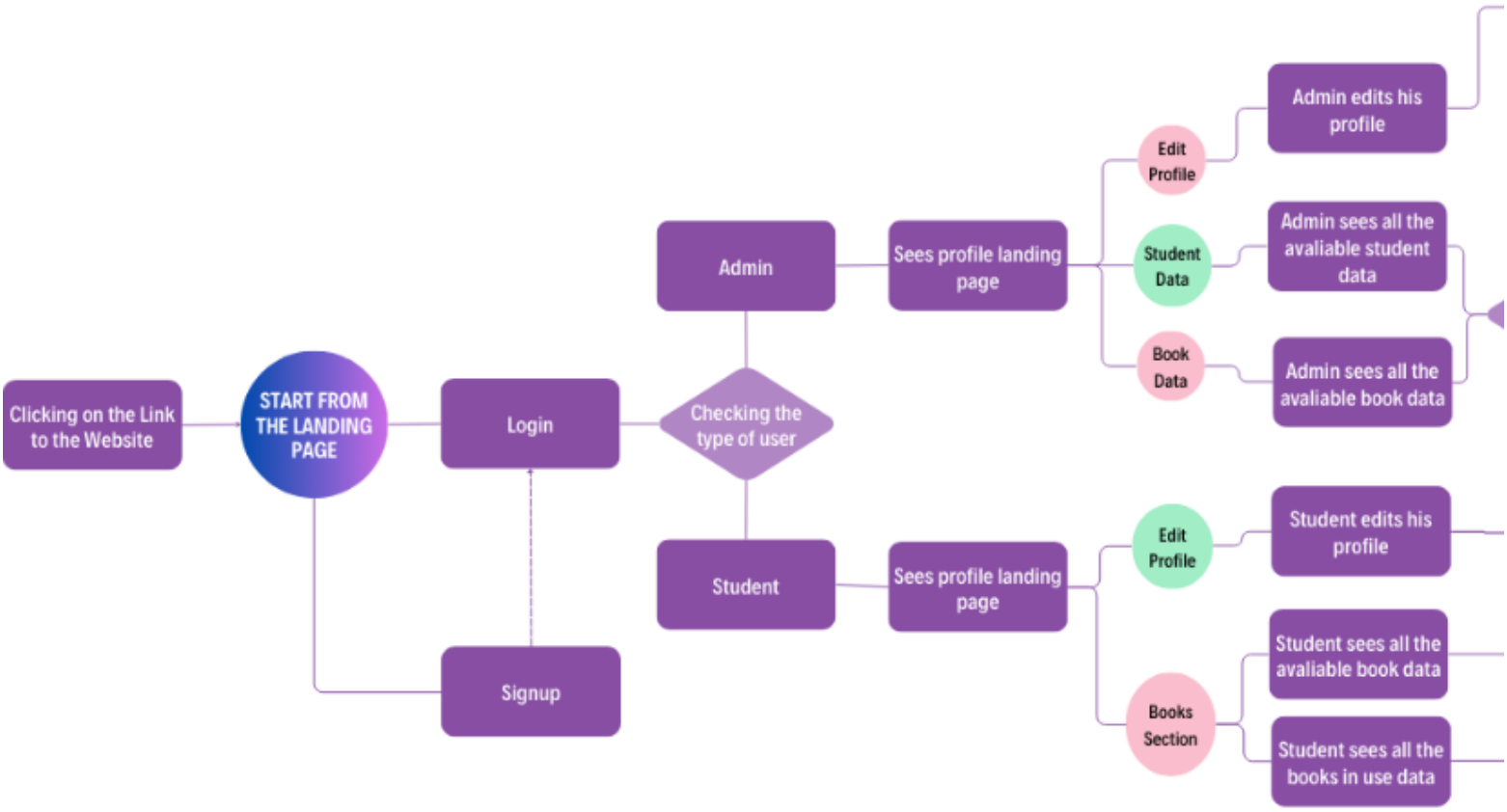
9. Book renting service (unless explicitly mentioned).

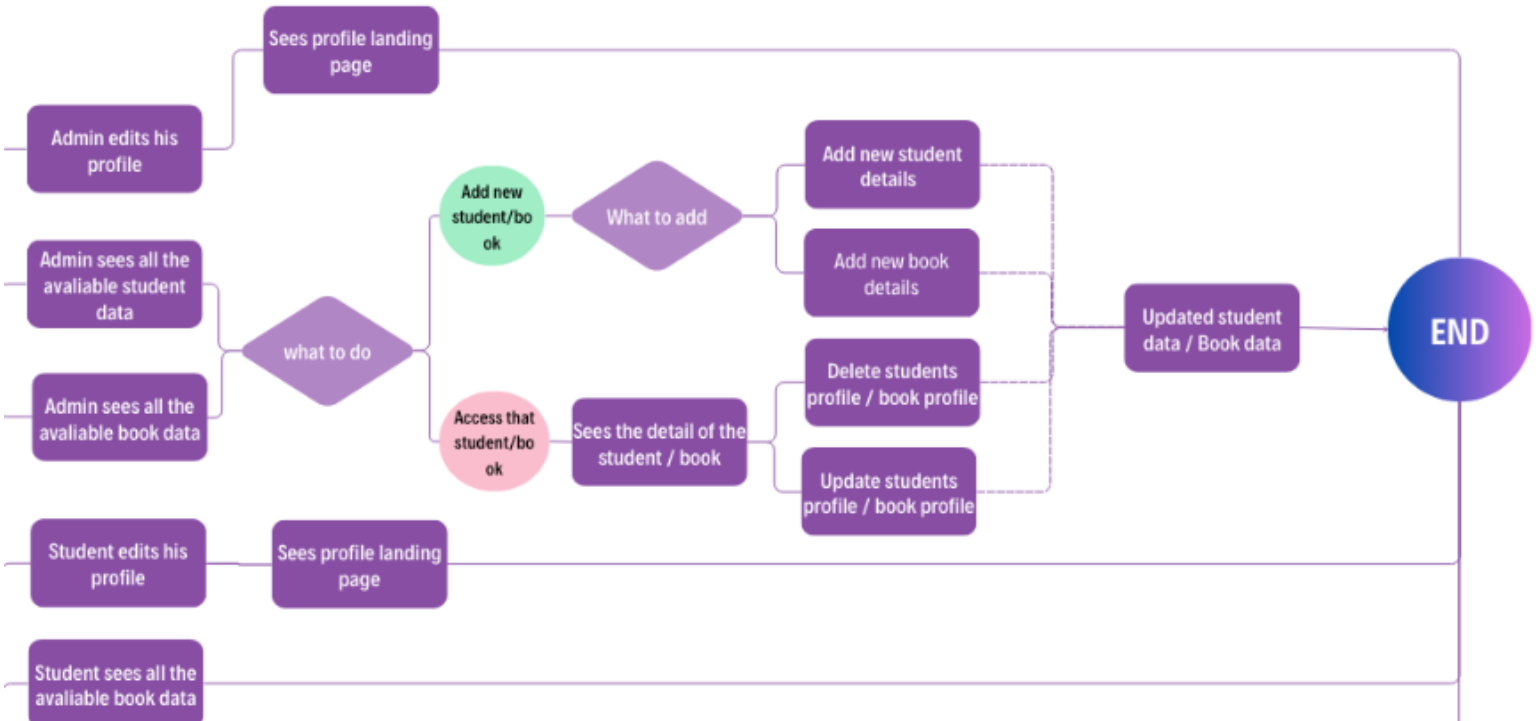
Note: All the features having "Unless explicitly mentioned" would be integrated in the updated version of the website, so as to properly test the functionality and efficiency of the current features.

Design

1. ***UX Diagram:*** ***User Flow Diagram Link***

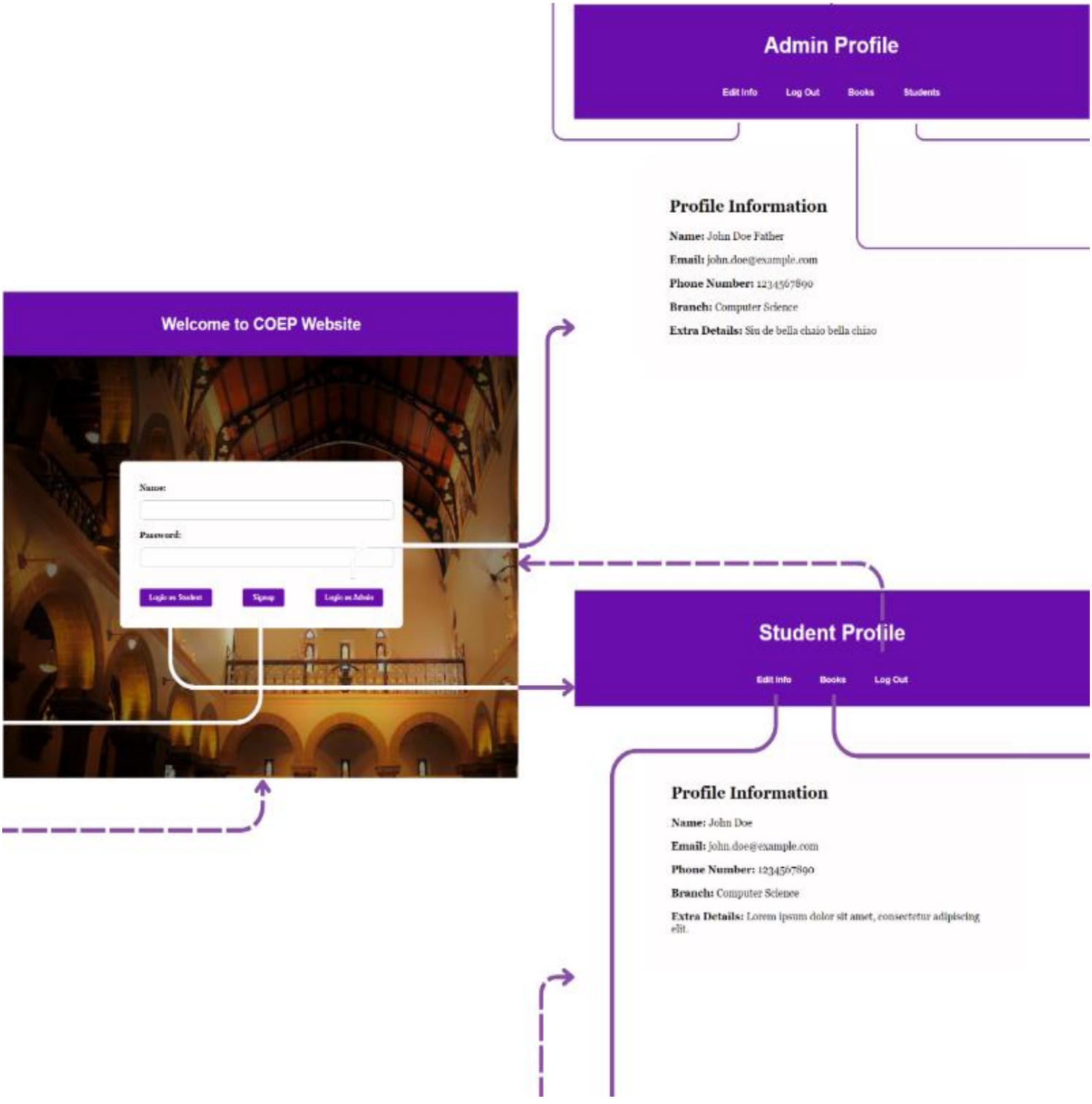
Student and Book Management System User Flow Diagram



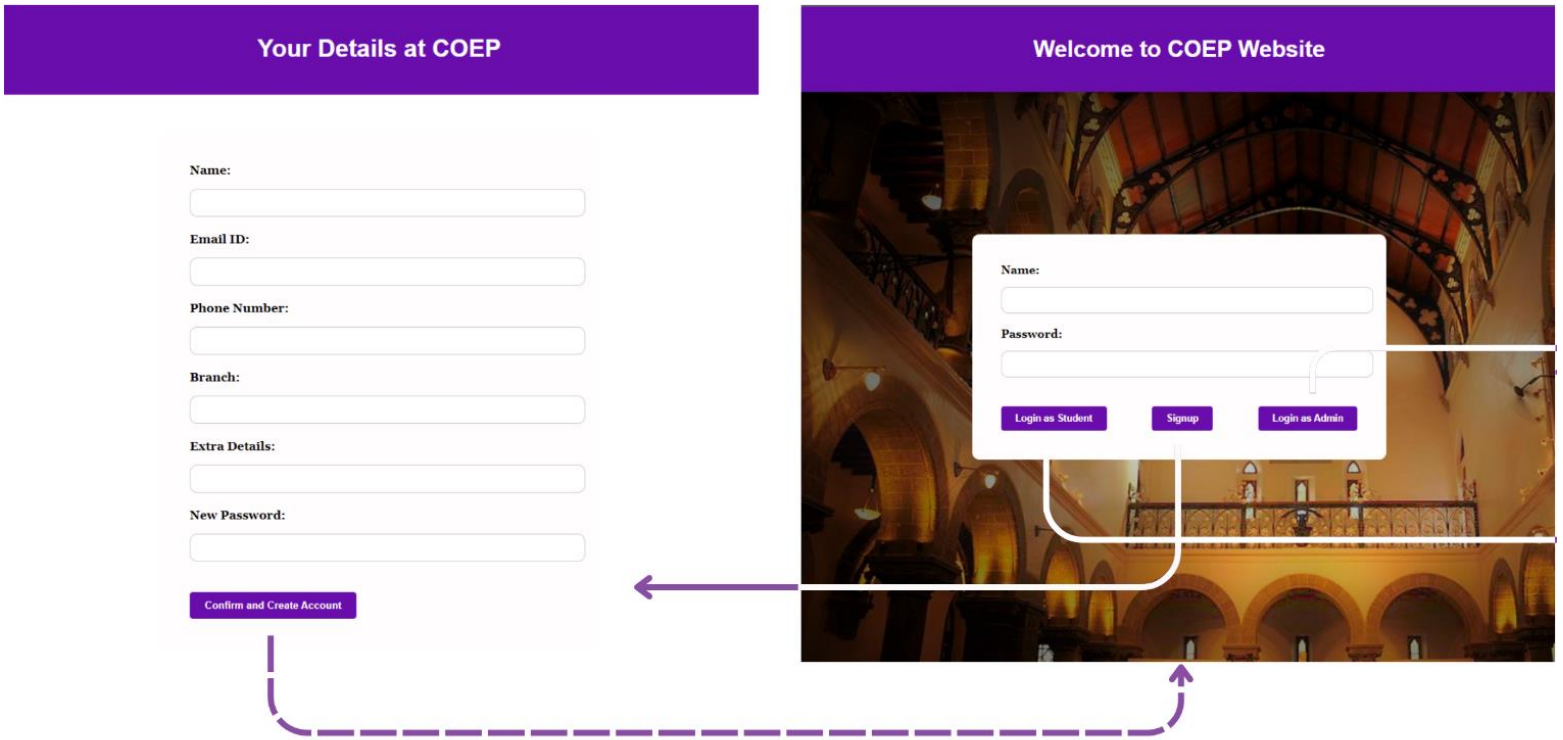


2. *UX Diagram: [UX Design Link]*

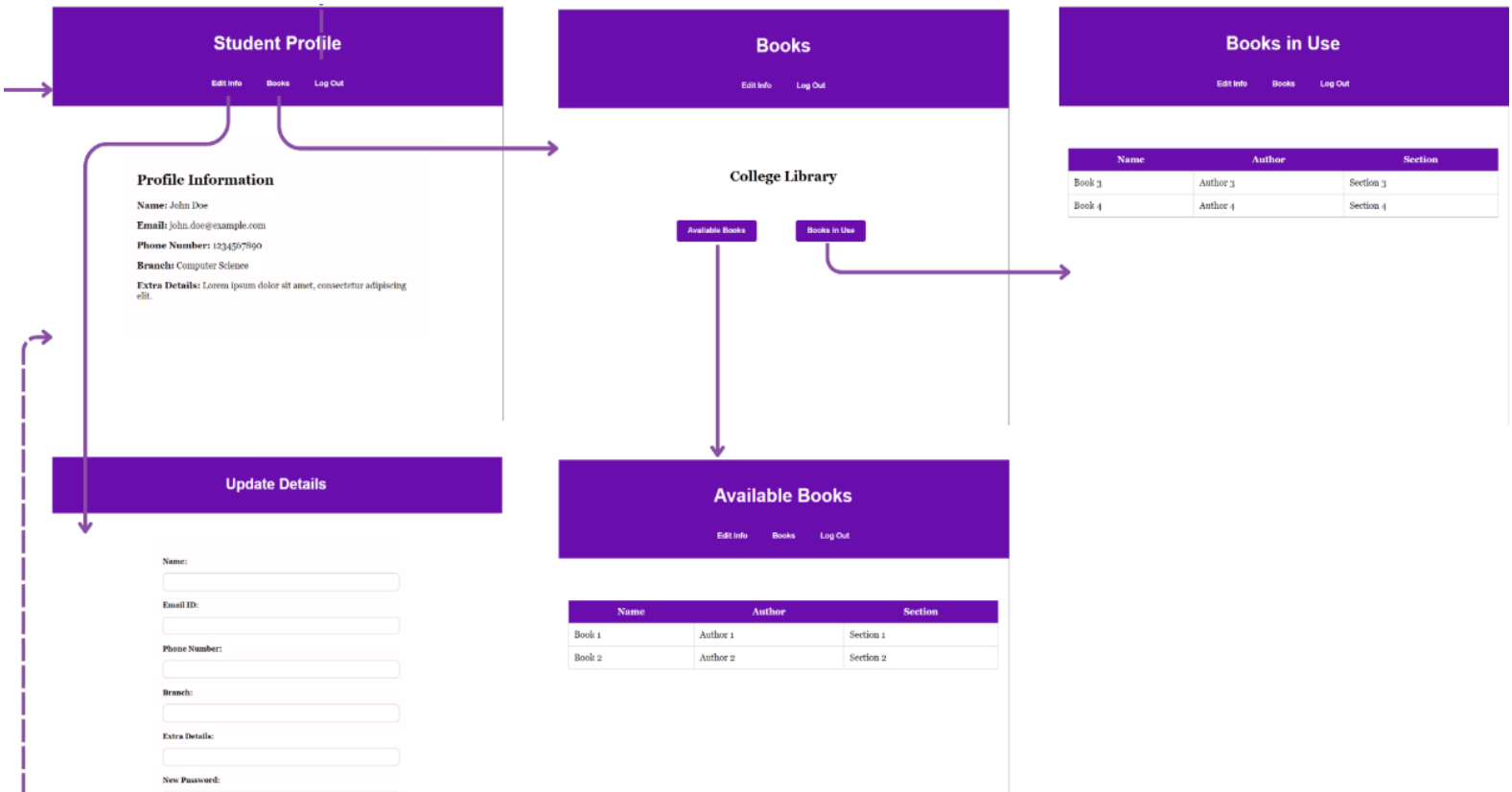
Landing page for the website and its access to the Admin and student's Landing page:



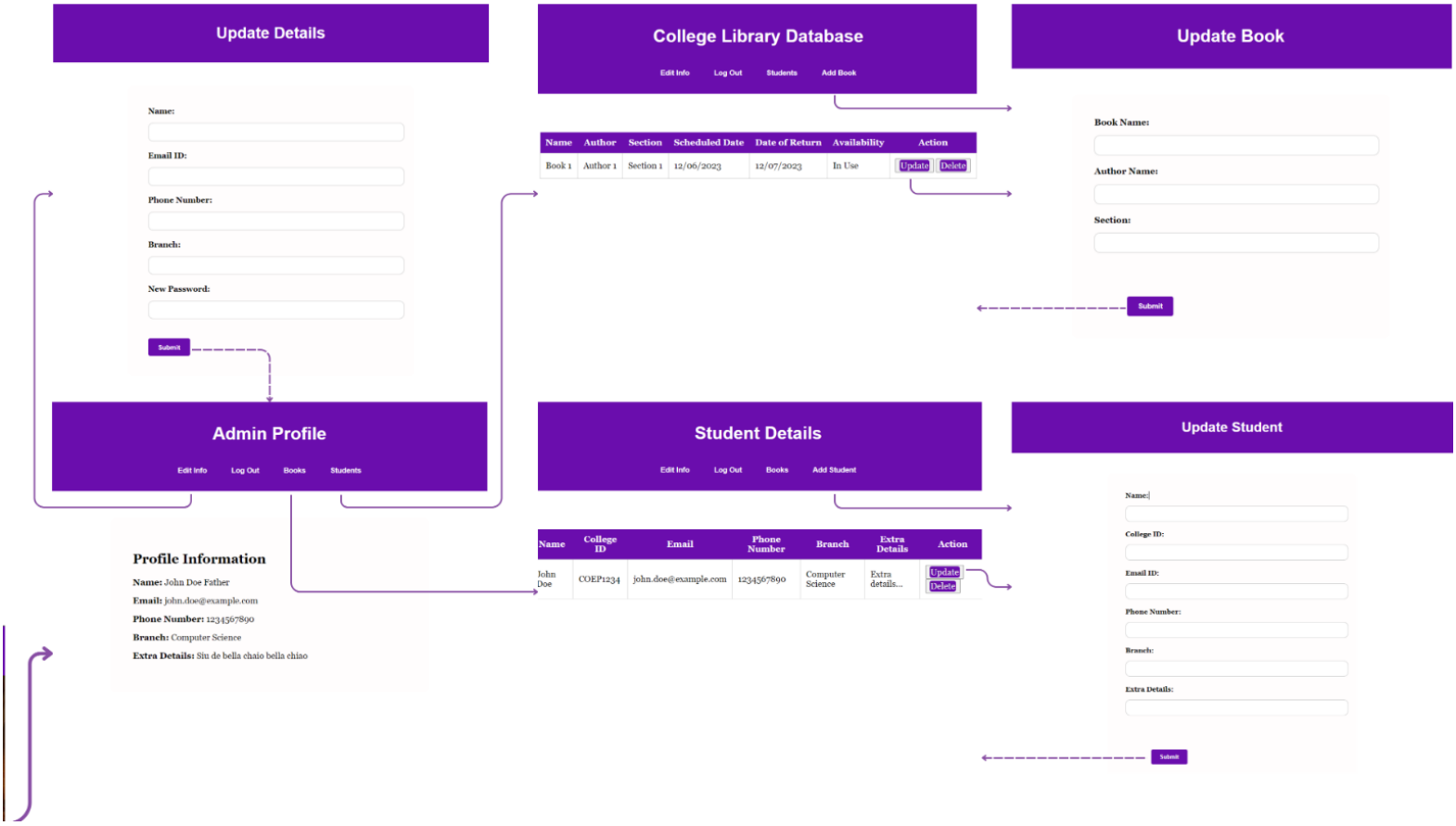
Creating account for new user:



Navigating through the functionalities of Student profile



Navigating through the functionalities of Admin profile



3. UX Design Principles/Guidelines/Standards

The design principles which we kept in mind while designing the UX were:

- Simplicity and Minimalism: So that the whole college could adapt to the simple yet effective interface by keeping the design clean, simple, and uncluttered to reduce cognitive load on users.
- User-Centered Design
- Consistency: Consistency is maintained across the whole website in design elements such as typography, colors, icons, and interaction patters to help users develop a mental model of system and improve usability of the product.
- Responsive Design: Ensuring that the system is accessible and usable across different devices and screen sizes

- Information Architecture: Organize and structure the information in a logical and intuitive manner. Use categorization, labeling, and hierarchical structures to help users easily find and navigate through the system.
- Performance and Speed: Optimize the system's performance to provide a fast and responsive experience. Minimize loading times, reduce latency, and optimize server-side processing.
- Visual Hierarchy is created across the website

Some other principles which will be implemented in the later versions of our product be:

- Dark Mode: Providing users with the option to switch to a dark color scheme, which will reduce eye strain and improve usability.
- Micro interactions: Small and subtle animations &interactions will be added to improve engagement and provide feedback.
- Personalization: The system would be tailored as such to allow users to personalize the user interface to cater their personal choices thus, increasing user engagement and satisfaction.

DBMS

1. ***Database Design***

   Our data base has 3 different schemas which are:

   - User Data – Used for login, signup, and students list
   - Admin Data – Used for authorization
   - Books Data – Used for books list and maintaining the virtual library

All the data mentioned above are individual primary keys with no inter dependency and thus no foreign key is present.

All have their individual functionalities and requirement and their respective schema is in the table given below;

USER	ADMIN	BOOKS
{ *_id: ObjectId,* *Name: String,* *email: String,* *branch: String,* *password: String,* *Extra Details: String* *}*	*{* *_id: ObjectId,* *Name: String,* *Email: String,* *branch: String,* *password: String,* *Extra Details: String* *}*	*{* *_id: ObjectId,* *Book Name: String,* *Book Author: String,* *Section: String,* *}*

2. Database Design Principles/Guidelines/Standards

Some of the database design principles which we kept in mind whike making the website are:

When it comes to database design principles, there are several established guidelines and standards that are commonly followed. While the specific design choices may vary depending on the requirements and nature of the data, here are some principles and ideas that are commonly used:

1. Normalization: Normalization is a technique used to eliminate data redundancy and improve data integrity in the database. We followed the normalization rules (such as the first, second, and third normal forms) to ensure that data is organized efficiently and avoid data inconsistencies.

2. Denormalization: Denormalization is the process of intentionally introducing redundancy into a database design to improve performance.

3. Data Integrity: Ensuring that appropriate data integrity constraints, such as primary key constraints, foreign key constraints, and unique constraints, are defined.

4. Data Security: Implementing proper access controls and permissions to ensure data security and prevent unauthorized access.

5. Scalability and Performance: Design the database schema and query patterns with scalability in mind.

6. Consistency and Concurrency

7. Documentation and Standards: Maintaining proper documentation of everything and adhering to best coding practices to improves the readability and extensibility of the code

3. API Design

As of now in our product, no integration of APIs was required as it was all done through other ways like;

- Instead of authentication and user management APIs our website can directly link with the data base to verify a user and thus proceed with the login.
- Instead of Student Details API, all the students of an institution would be required to sign up on the website which in turn would provide us with the data required.
- Instead of external Administrator API, the data base comes prefilled by the super-admin itself to facilitate the authorization of the administrative staff and is tightly monitored. Thus, use of external APIs reduces the security in all the above-mentioned cases.

Some other APIs which would be used by us in the future would be:

- Library Catalog API – So that instead of manually updating the book values we will be able to have direct access to the book details. This in turn would provide us with better data like Unique book token and other niche details about the respective book.

- Notification and Reminder API – For the later versions of our website, this API is very much required to easily integrate the website in day-to-day usage via notification and reminders. This notification feature would be used mostly for book lending and returning along with added benefits for the admins.
- Analytics and Reporting API – This would help us keep track of the behavior of students in terms of Book borrowing and also provide us with better analytics to improve the overall website.

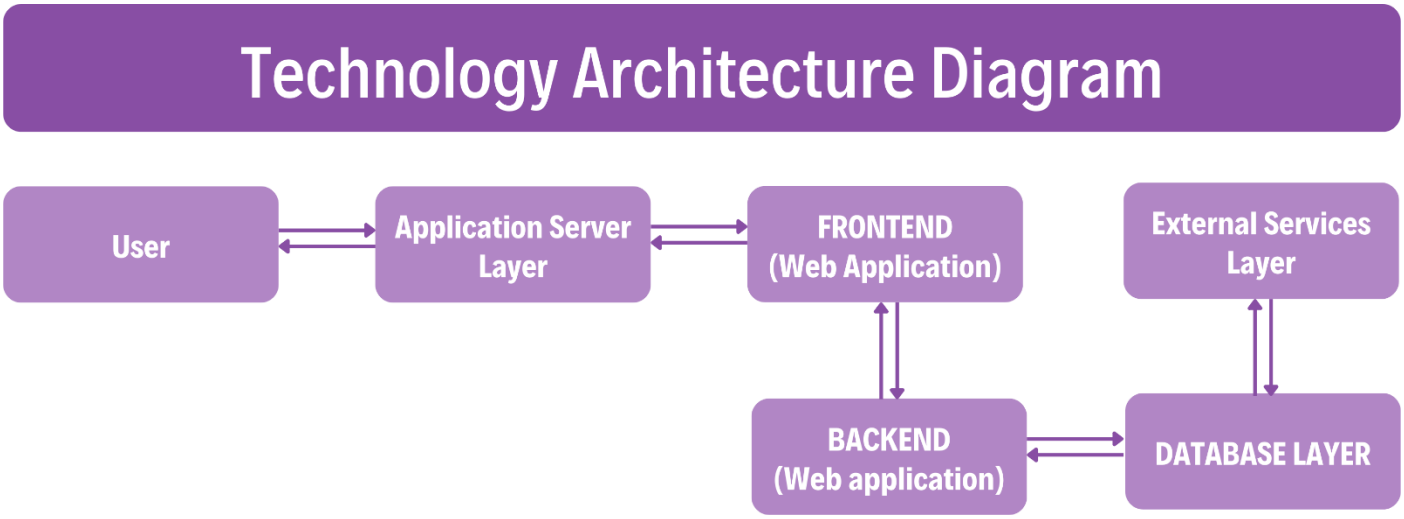
4. Provide API Design and Development

As mentioned above API design and development isn't required as of now.
Still, the steps which would be looked after in the upcoming versions would be:

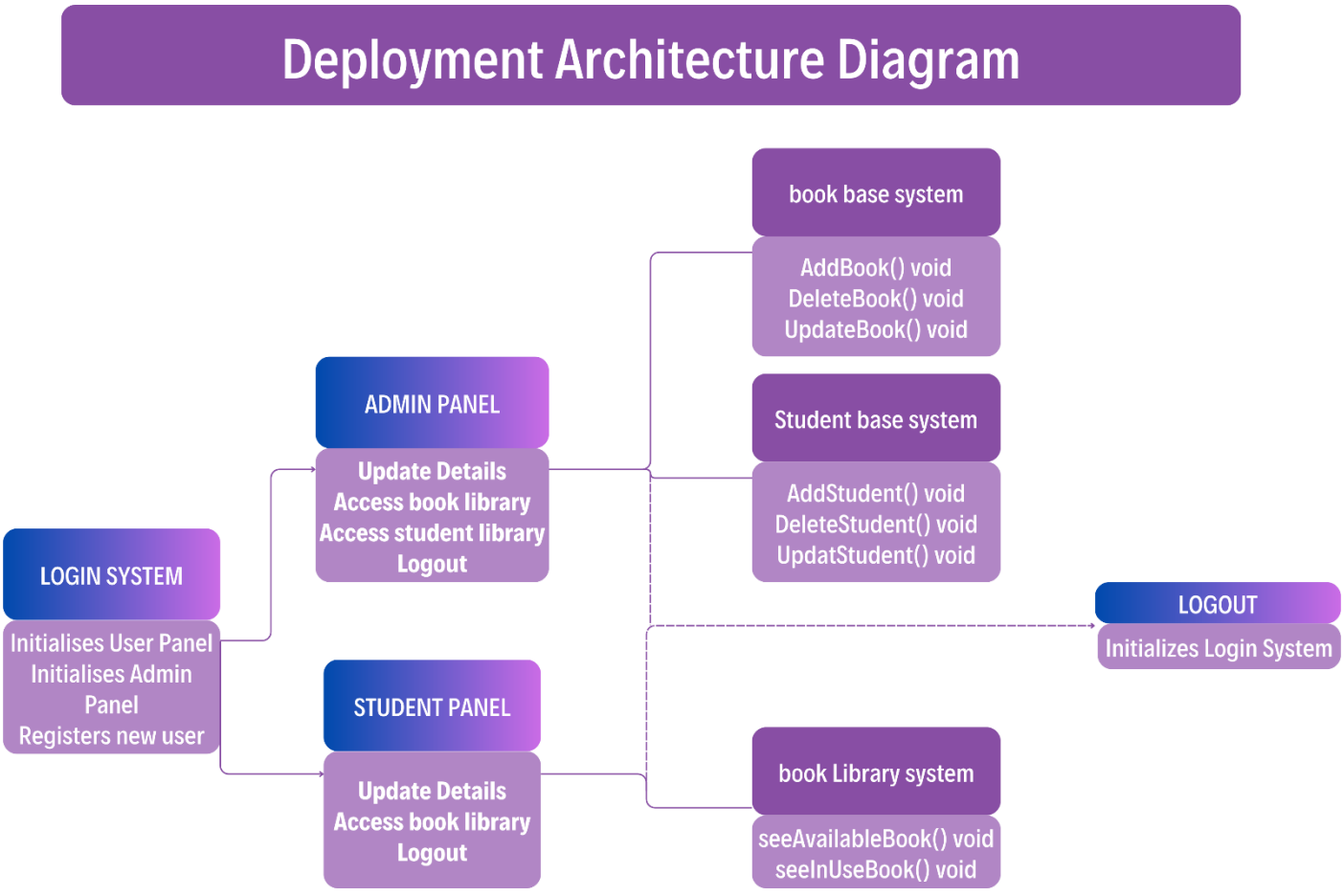
Use Cases and Functional Requirements
Choose the Right API Style/Architecture
Define API Endpoints and Data Models
Determine Authentication and Authorization Mechanisms
Design Request and Response Structures and thus implement API Endpoints
Develop the backend logic and functionality for each API endpoint
Test and Validate the chosen APIs and thus, Handle error cases and exceptions
Documentation and API Specification
Monitor and Improve over the same

Technology Architecture Design

1. Technology Architectural Component Diagram



2. Deployment Architecture Diagram



Other Features

1. ***Mention Technologies:***
    - MERN Stack Development using:
      MongoDB, ExpressJS, React and NodeJS.
    - MongoDB – for database
      ExpressJS and NodeJS – for backend server logic
      ReactJS- for frontend
    - HTML, CSS – for Web Page content and styling
    - Handlebars – for including Dynamic HTML elements

2. ***Mention Tools:***
    - VS code
    - Mongo Shell and Atlas Compass
    - Code Pen
    - Microsoft PowerShell
    - Canva
    - GitHub and GitHub Desktop

3. ***Scalability Features:***
    - Load balancing
    - 2 factor authentication
    - Dark mode
    - Notification Features
    - Book renting app
    - Horizontal scaling
    - Caching
    - Content Delivery Network (CDN)
    - Database scaling
    - Microservices architecture
    - Distributed file storage

- Monitoring and auto-scaling

4. ***Describe High Availability Features:***
    - Multizonal Deployment
    - Hybrid Cloud Deployment
    - Redundant Hardware
    - Automatic Recovery and data backup
    - Continuous Monitoring and Proactive Maintenance

5. ***Describe Security Features:***
    - Integration with third-party Identity and Access Management (IAM) systems
    - Role-Based Access Control (RBAC) system
    - Two-factor authentication (2FA) or multi-factor authentication (MFA)
    - Security monitoring and intrusion detection systems
    - Notifications for Security monitoring
    - Secure coding practices and vulnerability assessments
    - Regular security audits and penetration testing
    - Secure API design and implementation
    - Logging/Auditing Features

6. ***Describe Logging / Auditing Features:***
    - Date/Time stamping of events for accurate tracking
    - Detail logging of system activities associated with certain action or events
    - Centralized logging of data
    - Integration of proper security and event management information
    - Monitoring and alerting suspicious activities of anomalies
    - Logging of system errors, exceptions, warnings, etc.

7. ***Describe Monitoring Features:***
    - Resource Monitoring
    - Application Performance Monitoring (APM)
    - Server and Infrastructure Monitoring

- Network Monitoring
- Database Performance Monitoring
- User Activity Monitoring
- Error and Exception Logging

8. ***Notification Features:***

- Email Notifications
- Breach alerts
- SMS Notifications
- Monitoring and tracking activity
- In-App Notifications
- Deadline and reservation reminders
- Account Activity Notifications

Development

1. ***Source Code/Git :*** ***https://github.com/prajwal12vr/Student_and_Library_Base_management_system.git***

   In our case:
   Full-stack, Documentation, Devops and Testing repositories are present inside one main repository of SEL-BMS

   There are many repositories like:

   - Front-End repository- Contains the code necessary for the UX of the product
   - Back-end repository- Contains the code responsible for server-side logic, data processing and interactions with databases or external APIs
   - Full-stack repository- Contains both the front end and backend logic
   - Mobile-app repository- Contains the codebase for Ios and Android applications

- API repository-Contains the code responsible for creating and maintaining the application programming interfaces (APIs) that allow communication between different software systems
- Devops repository- Contains the infrastructure and deployment related code
- Testing repository- Contains the code and resources for testing the application to ensure the quality of product and fix any bugs using the test scripts present inside
- Documentation repository- Contains project related documentation, guidelines, use case information, manuals and other miscellaneous data
  Etc.

2. Branching strategy

Our branching strategy was pretty straightforward.

It was a 2-person project and thus along with our Main branch two sub branches of "*Praj"* and "*harsh"* were created where we pushed our respective progress only when it seemed fit via GitHub Desktop.

When any change in the website was final and both the teammates agreed upon the same, then only with the allowance of both the members could the code changes be pushed into the *"Main"* branch.

This strategy safeguarded us from any unanticipated blunder.

3. Setting up Developer Machine

1. Firstly, import the file by forking it from the GitHub repository
2. Open the folder on VS Code
3. Go to the package.json file
4. Open new terminal on VS code
5. Type "npm install" to install all the dependencies (you will see a new folder named as node_modules created thereafter)
6. Download MongoDB from online and install it in your device
   While installing do mark yes to the installing of MongoDB compass

7. Lastly go again on the terminal and click on “npm start”
8. You will see the following message as;
   *Server running on **http://localhost:3000***
   Connected to MongoDB
9. Ctrl + click on the link which you get above or directly go to the localhost:3000 on your browser
10. You will thus see a fully functioning website

    Thankyou!

4. Unit Testing Method

Unit Testing has been done for each and every individual unit of the website based on the deployment architecture diagram given above which contains all the containers and clusters required for testing.
Extensive manual unit testing has been done on the same.

Testing

1. Test Scope

We have tested al the features which are in Scope as mentioned in the above document.

Also, the website was given for user integration testing and the results came out to be positive due to the smooth, user friendly UX design.

Since it is a lightweight website all the testing was done manually to check the different test cases which are the various functionalities of the website as mentioned in the Screen flow Diagram.

As a developer direct unit testing was done at each functionality to test our product.

2. BVT Test Cases

As mentioned above, User integration testing and along with that separate user acceptance testing was done.

We did live probing and happy-go-path were tested.

Feedbacks were taken to improve the website.

UX design was constantly integrated to fit in the feedbacks and the functionalities

In this way BVT aka Business verification testing was successfully done.

3. ***Test Automation Objective & Scope***

   Test Automation Objective is to automatically set testcases and perform testing on them for ease and efficiency.

   In our case, it is not in scope as we have a lightweight application and thus, manual testing takes same or even less amount of time that automated testing.

4. ***Test Strategy / Test Plan***

   The test strategy was to check each and every functionality of the website manually.

   At the same time live monitoring of Data input, output and update was done through MongoDB compass and VS Code live server.

   Thus, overall the testing went smoothly.

5. ***Test Automation Strategy / Plan***

   As mentioned above the current website didn't have automated tests and thus no automation strategy is present but it is indeed a priority for the later versions of the website due to the integrated scaling which will be undertaken.

6. ***Test Automation Branching Strategy***

   As stated above, it would be included in the later release.

7. ***Test Code/Git***

As mentioned above, manual testing was done due to its greater efficiency in the scope of our website and thus a test code for the same is not present.

DevOps

1. ***Dev Integration Environment Set up***

   Nexus and Jenkins pipelines for local deployment would be used.

   Thus, the infrastructure provisioning steps would be kept in mind to set up the Dev Integration Environment using the CI+CD pipeline instructions.

2. ***QA Environment Set up***

   Qualitative analysis is not set yet because the system was tested locally and the deployment was tested on the live server.

3. ***Staging Environment Set up***

   Unlike a banking application which needs to be up and running continuously and should be monitored likewise, ours is a lightweight database management system which needs to be staged in the appropriate environment but live like set up is not necessary.

   Which means in case of testing, updates or validation the system can be closed down for few hours without the requirement of a staging environment or live like cloning.

4. ***Backup for DevOps and PII production data masking***

   - Identify data to backup
   - Choose backup method
   - Define backup schedule
   - Implement backup solution and perform backup
   - Verify backup integrity
   - Document and retain backup policies

   - Identify Sensitive Data

- Define masking techniques
- Develop masking strategy, scripts and processes
- Test and validate masked data
- Document the data masking procedures

Production Deployment

1. Infrastructure Setup:

- Provision the necessary infrastructure components, such as servers, networks, and storage, based on the system requirements.
- Install the operating system and required dependencies on the servers.
- Configure network settings, including IP addresses, subnets, and firewall rules.
- Set up any load balancers, DNS configurations, or proxy servers as needed.

2. Application Deployment and Configuration:

- Prepare the application for deployment by packaging it into a deployable format, such as a WAR or JAR file.
- Copy the application package to the production environment.
- Install and configure any runtime environments or frameworks required by the application.
- Configure application-specific settings, such as database connections, API integrations, and environment-specific configurations.

3. Logging/Auditing Configurations:

- Choose a logging framework or tool that suits your application's needs.
- Configure the logging framework to define log levels, log formats, and log destinations (e.g., file or database).
- Set up auditing mechanisms to track and log relevant events, actions, or changes in the system.
- Define the retention period for logs and establish log rotation policies.

4. Scalability Configurations:

- Determine the scalability requirements for the system, such as horizontal scaling or load balancing.
- Configure load balancers or clustering mechanisms to distribute traffic across multiple servers.
- Set up auto-scaling rules to automatically provision or deprovision resources based on demand.
- Implement caching mechanisms to improve performance and scalability.

5. Security Configurations:

- Enable encryption for data in transit and at rest using SSL/TLS certificates and encryption algorithms.
- Implement access controls and authentication mechanisms, such as username/password or Single Sign-On (SSO).
- Set up security measures like firewalls, intrusion detection systems, and security patches.
- Implement secure coding practices, input validation, and output encoding to prevent common security vulnerabilities.

6. Credentials:

- Identify the appropriate Business Unit (BU) and IT Points of Contact (POC) responsible for managing credentials.
- Define the frequency for changing credentials, such as passwords or access keys, based on security best practices and organizational policies.

7. Certificates:

- Determine the required certificates, such as SSL/TLS certificates for secure communication.
- Identify the certificate issuing authority (CA) and obtain the necessary certificates.
- Download the certificates from the CA's website or certificate management platform.
- Install the apt servers following the CA’s instructions and guidelines.

Production Testing

Whenever doing production testing it is important to disconnect it from the actual server and product by making a clone of it.

At the same time, proper data backup and security measures should be taken to prevent any hindering circumstances.

Lastly all the dependencies and system programs should be properly ensured in working to further the production testing.

Backup and Recovery

Backup and Recovery currently happens via the safe storage and branching of data in the local Server

Monitoring

1. ***The list of Key parameters to be monitored are:***
    - Monitor the uptime and availability of system to ensure its accessibility to the users
    - Monitor the server health metrics such as CPU Usage, memory utilization, disk space and network bandwidth
    - Database performance
    - User authentication and access
    - Proper maintenance of database, library catalog and student data
    - Student activity
    - System logs and error tracking
    - Security features
    - Backup and disaster recovery
    - Scalability, updates and resource planning
2. ***List of emails and aliases to notify:***
    - Prajwal (super-admin) – workpvr@gmail.com

- Harsh (super-admin) – harshbihani@gmail.com
- Administrators – (to be decided by the institution)

Routine Maintenance

1. ***The areas for routine maintenance for our website are:***

- Database maintenance:

  Majority of our website revolves around Database management and thus its routine maintenance is a key concern.

  The database used in our system is NoSQL aka MongoDB and thus, regular backups of the database is necessary to ensure data integrity.

  Analyze database performance using monitoring tools for the ever-increasing footfall of the website and also identify areas for optimization.

  Plan for additional storage if required.

  Perform periodic database health checks.
- Security Updates:

  Stay informed about any security vulnerability to keep the database of the system safe.

  Strict monitoring of Admin Logins to prevent any data breach in the system.

  Up-to-date data base maintenance

  Fix any security patches and test the system thoroughly after the updates to ensure security and functionality.
- Performance Monitoring:

  Use performance monitoring tools to collect data on system resource utilization and analyse the overall performance of the websites metrics such as CPU Usage, memory allocation and utilization and I/O parameters functionality.
- Backup and Recovery:

  Constantly update your database recovery systems inline with your product requirement to ensure data insurance in case of any emergency, data breach or total system failure.

2. ***Other dependencies renewal:***

   To ensure that the internal dependencies and components remain up-to-date, secure and compatible; we can take the following measures:

   - Renewal of generic components.
   - Identify internal dependencies and keep them updated
   - Track version and release information
   - Test in a non-production environment
   - Thorough occasional testing
   - Plan and schedule updates
   - Post-Update validation
   - Proper documentation and knowledge sharing
   - Regular maintenance

Emergency Maintenance

1. For Emergency Maintenance Details you can contact:

   - Harsh Bihani - 78881 36562
   - Prajwal Randive - 7249657113

2. The list of emergency use cases and the troubleshooting hints for the same are:

   - Emergency Use Case 1: System Downtime

     Troubleshooting Hints:

     1. Check server and network status

     2. Examine system logs

     3. Database connectivity

     4. Load and performance

- Emergency Use Case 2: Data Corruption or Loss

    Troubleshooting Hints:

        1. Backup and restore

        2. Data integrity checks

        3. Recovery from transaction logs

        4. Preventive measures

- Emergency Use Case 3: Security Breach or Unauthorized Access

    Troubleshooting Hints:

        1. Assess the scope of the breach

        2. Review access controls and user permissions

        3. Investigate security logs

        4. Implement security patches and measures

Marketing Support

Given below is the link to the document to market our given product
Marketing Support Flyer