Project Phase 2

Ahmad Ghosn, Amal Al Howry, Yasmine Ismail, Dema Ben Jabr, Julio Costa Sanabria

Ranking Criteria for Use Cases:

a. User Importance: How critical is the use case for the primary users of the application, i.e., patients, doctors, and administrative users?

b. Impact on Patient Care: Does the use case directly impact the quality of patient care and their overall experience using the application?

c. Efficiency and Productivity: How much does the use case improve the efficiency and productivity of the healthcare providers and the administrative staff?

d. Regulatory and Compliance: Does the use case help the application comply with healthcare regulations and privacy standards?

e. Scalability: Does the use case allow for future scalability and growth of the application?

f. Competitive Advantage: Does the use case provide a competitive advantage in the healthcare software market?

Rank of the Used Cases

1. Appointment Booking: This use case is highly important for patients and healthcare providers, as it directly impacts the scheduling of appointments, which is critical for patient care. It also improves efficiency and productivity, complies with regulations, and can provide a competitive advantage.

2. Admin Dashboard: Administrative users play a key role in managing the system, user accounts, and performance. A well-designed admin dashboard is essential for the overall functioning of the application and regulatory compliance.

3. Patient Registration: Patient registration is the foundation of the application, and it is important for user acquisition. It directly impacts the user experience and is crucial for compliance.

4. Provider Scheduling: This use case is important for doctors and nurses, but its impact on patients' direct experience is somewhat indirect. However, it is essential for healthcare provider efficiency.

5. Feedback and Reviews: While patient feedback is important for continuous improvement, it doesn't have as immediate an impact on patient care or regulatory compliance as other use cases.

6. Appointment Reminders: While important for patient engagement, appointment reminders are less critical than other use cases for the overall functioning of the application.

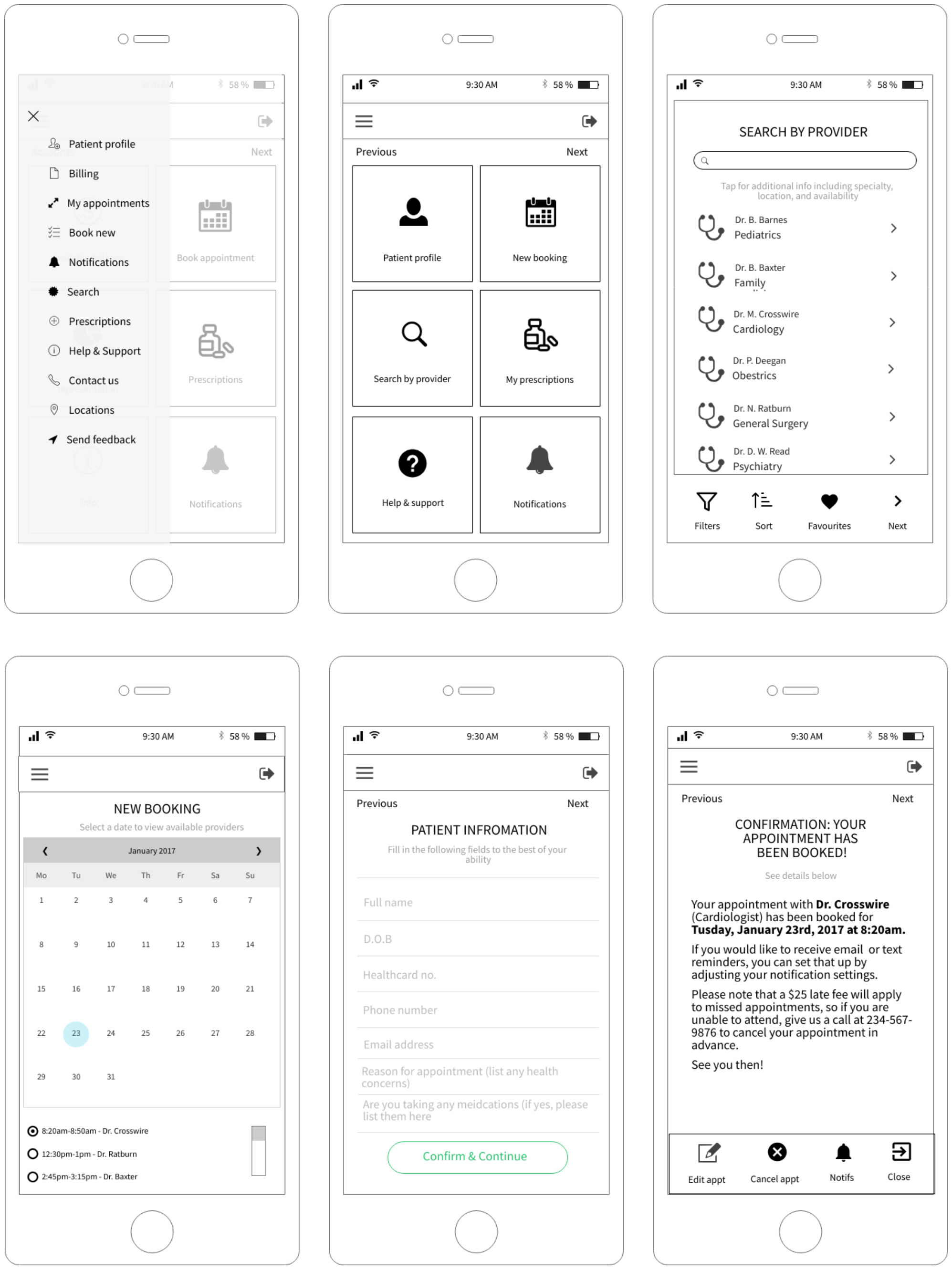
Detailed Description and UI Sketch for the 3 Most Important Use Cases:

1. Appointment Booking:

Use Case Description: Patients can search for and book appointments with available doctors and nurses. This feature should provide a user-friendly interface to browse available healthcare providers, view their schedules, and select suitable appointment slots. Patients can also provide information about their medical condition to help match them with the right provider.

UI Layout Sketch:

- A search bar to find healthcare providers.
- A list of available providers with their names, specialties, and availability.
- A calendar view to select appointment dates and times.
- A form to enter patient details and medical history.
- A confirmation page with appointment details.



Your appointment with **Dr. Crosswire** (Cardiologist) has been booked for **Tusday, January 23rd, 2017 at 8:20am.**

If you would like to receive email or text reminders, you can set that up by adjusting your notification settings.

Please note that a $25 late fee will apply to missed appointments, so if you are unable to attend, give us a call at 234-567-9876 to cancel your appointment in advance.

See you then!

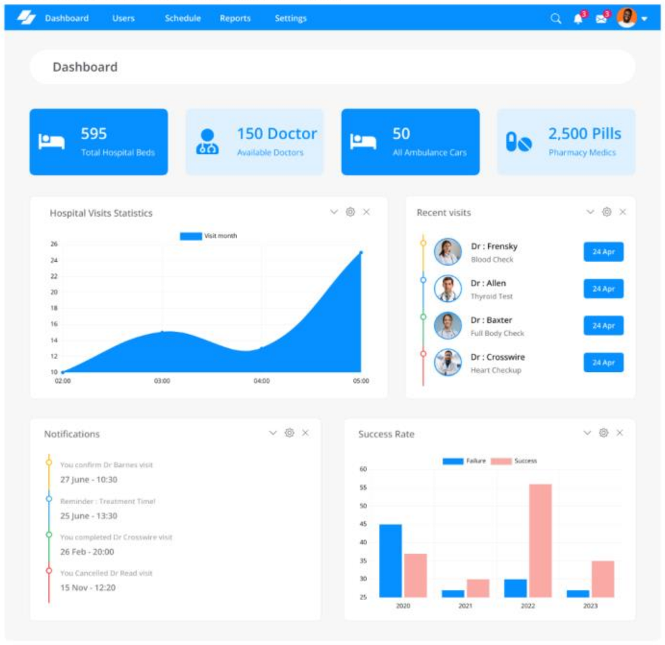
2. Admin Dashboard:

Use Case Description: Administrative users can manage system settings, user accounts, and monitor overall system performance. The dashboard should allow administrators to add or remove healthcare providers, manage user accounts, set system preferences, and access performance metrics and reports.

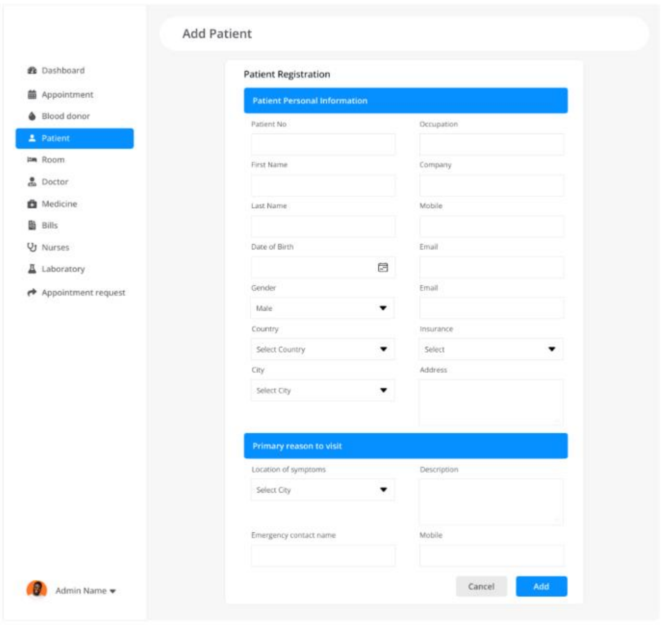
UI Layout Sketch:

- A dashboard homepage with an overview of system performance.
- Navigation menu with options to manage users, settings, and reports.
- User management interface for adding, modifying, and removing user accounts.
- Settings page for configuring system preferences and notifications.
- Reports section with charts and data tables for monitoring system usage and performance.

Dashboard homepage + navigation menu + reports summary + settings preview



Add/remove/modify users (patient shown below, Doctors/Nurses/Other providers will be roughly identical)

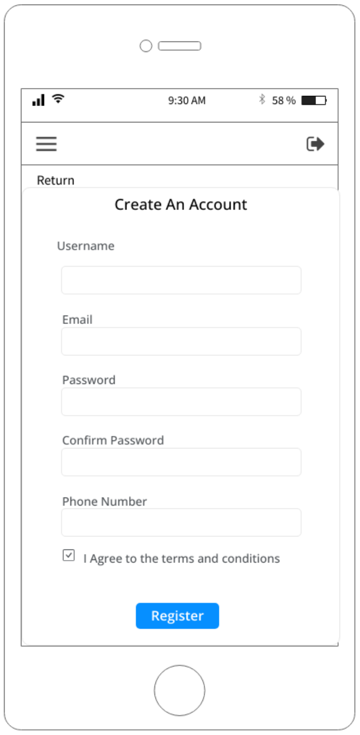


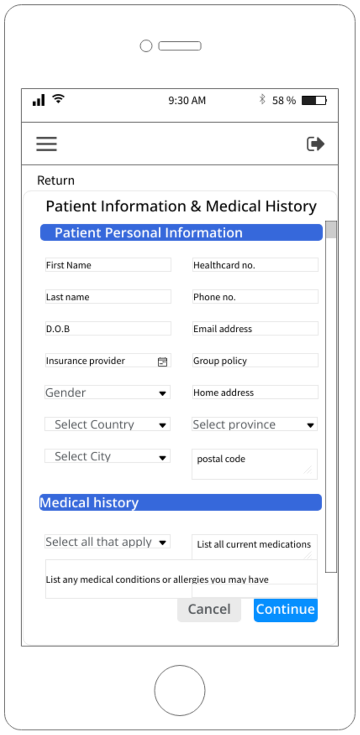
3. Patient Registration:

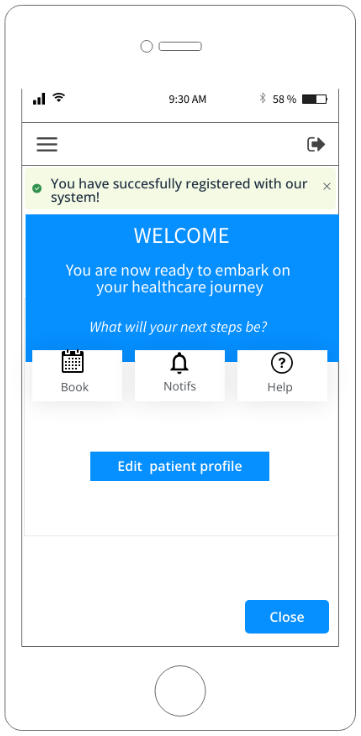
Use Case Description: Patients can create accounts, providing necessary personal and medical information. This process should be simple, secure, and compliant with healthcare data privacy regulations.

UI Layout Sketch:

- A user registration form with fields for personal information (name, contact details) and medical history (allergies, previous surgeries).
- Secure account creation with password and account verification steps.
- A consent form to ensure compliance with privacy regulations.
- Confirmation page welcoming the patient to the system.







You have succesfully registered with our system!

You are now ready to embark on your healthcare journey

What will your next steps be?

<u>System requirements associated with these use cases</u>

1. Appointment Booking:

<u>Functional Requirements:</u>

User Authentication:

- The system shall implement a secure user authentication mechanism for patients and healthcare providers.

Provider Availability:

- The system shall dynamically display the availability of healthcare providers based on their schedules.

Appointment Booking:

- Patients shall be able to select a suitable time slot from the available options.
- The system shall confirm and notify the patient and healthcare provider of the booked appointment.

Patient Information Input:

- The system shall allow patients to input relevant medical information during the booking process.

<u>Non-functional Requirements:</u>

Performance:

- The system response time for appointment booking requests shall not exceed 2 seconds.

Scalability:

- The system should be designed to handle a minimum of 1000 concurrent appointment booking sessions.

Security:

- Patient information must be encrypted during transmission and storage.

- Access to healthcare providers' schedules should be role-restricted.

2. Admin Dashboard:

Functional Requirements:

User Management:
- The system shall provide administrators with the ability to add, modify, and remove user accounts.

System Settings:
- Administrators shall be able to configure system preferences, such as notification settings.

Performance Monitoring:
- The dashboard shall display real-time performance metrics, including user activity and system response times.

Non-functional Requirements:

Responsiveness:
- The admin dashboard shall load within 3 seconds.

Scalability:
- The system shall support at least 5 simultaneous administrative user sessions.

Security:
- Access to the admin dashboard shall be role-based and require multi-factor authentication.

3. Patient Registration:

Functional Requirements:

User Registration Form:
- The system shall provide a user-friendly registration form for patients to input personal and medical information.

Account Verification:
- Patients shall receive a verification email with a secure link for account activation.

Consent Form:
- The system shall include a consent form during the registration process for compliance.

Non-functional Requirements:

Usability:
- The registration process shall be intuitive, taking no more than 5 minutes for completion.

Security:
- Patient data shall be stored securely, complying with healthcare privacy regulations.

Performance:
- The registration process shall not exceed a 10-second response time for any user action.

This detailed representation outlines both functional and non-functional requirements for each use case in a format inspired by UML and natural language.

Architecture Description:

Overview: The proposed architecture for implementing the features in Project Phase 2 will be based on a modular and scalable design, following a microservices architecture. This approach enables flexibility, maintainability, and scalability, essential for accommodating the growing needs of the healthcare application.

Components:

1. **Frontend:**
   - Developed using a modern frontend framework (e.g., React, Angular) for a responsive and intuitive user interface.
   - Implements a RESTful API to communicate with the backend.
2. **Backend:**
   - Microservices Architecture: Utilizes independent microservices for Appointment Booking, Admin Dashboard, and Patient Registration. This allows for modular development, easier maintenance, and scalability.
   - **Appointment Booking Microservice:**
     - Manages appointment scheduling, availability, and patient-provider matching.
     - Integrates with external systems for healthcare provider availability.
   - **Admin Dashboard Microservice:**
     - Handles administrative tasks, user management, system configuration, and performance monitoring.
     - Utilizes secure APIs for communication with the frontend.
   - **Patient Registration Microservice:**
     - Manages patient account creation, information storage, and compliance with privacy regulations.
     - Integrates with authentication services for secure account creation.
3. **Database:**
   - Each microservice has its own database (e.g., MongoDB, PostgreSQL, MSSQL) for independent data management.
   - Ensures data consistency and isolation between different functionalities.
4. **Authentication and Authorization:**
   - Implements a robust authentication mechanism (e.g., OAuth) for secure user access.
   - Role-based access control (RBAC) ensures that users have appropriate permissions.
5. **API Gateway:**
   - Acts as a single-entry point for all microservices, managing requests and routing them to the respective services.
   - Enhances security and simplifies the frontend-backend interaction.
6. **Scalability:**
   - Each microservice can be independently scaled based on the specific usage patterns.
   - Cloud infrastructure (e.g., AWS, Azure) supports auto-scaling for efficient resource utilization.
7. **Monitoring and Logging:**

    - Incorporates monitoring tools to track system performance, identify issues, and ensure optimal functionality.
    - Logging mechanisms capture relevant events for auditing and debugging.

Justification:

- **Scalability:** The microservices architecture ensures scalability, allowing each component to scale independently, preventing bottlenecks during peak usage.
- **Modularity:** The modular design enhances maintainability, allowing updates or additions to one feature without affecting the entire system.
- **Security:** Role-based access control and secure communication protocols ensure the confidentiality and integrity of sensitive healthcare data.
- **Flexibility:** The architecture supports future feature additions or modifications without significant system-wide changes.