Software Requirements Specification (SRS) Document

Clinical Rostering for Hospitals
Team 4
Anushka Jain, Nandini Maroo, Utsav Shekhar, Md Faizal Karim

Brief problem statement

Hospitals are an essential service running 24/7 and require an adequate number of nurses and physicians to ensure that the best care facilities are provided to all patients. However, it is difficult to provide an efficient and fair schedule for the same manually. The Clinical Rostering project is meant to deal with this problem. This project involves developing a web application which accepts data and constraints regarding the availability of all the nurses and physicians, analyzes the data and, with the help of various scheduling and fairness algorithms, provides the most optimized schedule for the end-users.

System requirements

We'll be using React for the frontend , Storybook for UI development, PostgreSQL for the database, FastAPI for the backend and Optaplanner for designing the algorithm for shift allocation.

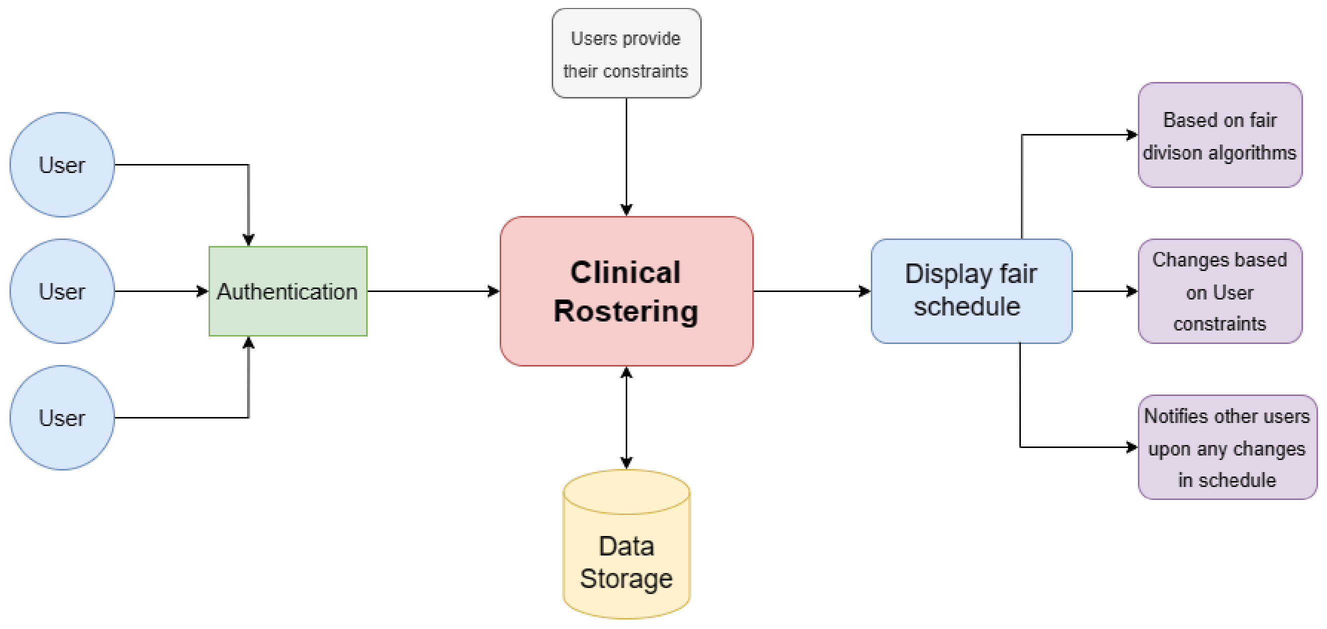
Users profile

The system would be used by nurses in the hospitals. It can be extended to be used by physicians, pharmacists, dieticians, and other hospital staff. The set of users mentioned have busy schedules thus the interface needs to be easy to use and access, concise and shouldn't take up a lot of time. The UI would be user-friendly keeping in mind that the users might not be very proficient with technology.

Feature requirements (described using use cases)

No.	User Case Name	Description	Release
UC-01	User Authentication	User sign in/registration to give access to modify/view data.	R1
UC-02	Employee Registration	A page to register employees (nurses, physicians, etc.) who work at the hospital along with their specializations, personal hard/soft constraints, etc.	R1
UC-03	Admin privileges	Admin login and option to accept/reject pending leave requests of employees.	R1
UC-04	Default schedule page	This page would display the current schedule of the employee - shifts and their timings.	R1
UC-05	Leave requests	The constraints including preferred shifts and specializations are added.	R1
UC-06	Providing the constraints to Optaplanner	OptaPlanner works behind the scenes by fusing powerful Artificial Intelligence optimization algorithms with highly effective score calculation and other cutting-edge constraint resolution methods.	R2
UC-07	Rendering generated schedule	The schedule obtained as a result of running OptaPlanner is updated on a webpage on our website.	R2
UC-08	Adding dynamic changes	Constraints can be modified in order to dynamically change the schedule.	R2
UC-09	Personalized notifications and google calendar integration	Users can see their schedule	R2

Use case diagram



Users provide their constraints
User
User
User
Authentication
Clinical Rostering
Data Storage
Display fair schedule
Based on fair divison algorithms
Changes based on User constraints
Notifies other users upon any changes in schedule

Timeline

Milestone	**Stipulated time**
Template designing	2 week
Authentication	2 week
Collection of data and storage in the backend	3 week
Integration of the app with Optaplanner	3 week
Addition of extra features.	1 week
Testing and bug fixes	1 week

Work Division [This is for the first 4 weeks]

Name	**Work**
Anushka Jain	Database
Md Faizal Karim	Backend
Nandini Maroo	Designing
Utsav Shekhar	Frontend

Feature Description

Use Case Number:	UC-01
Use Case Name:	User Authentication
Overview:	User sign in/registration to give access to modify/view data.
Actors:	User
Pre condition:	None
Flow:	Once a user is signed in, they will be redirected to the homepage from where they can access options to modify employees' data, modify constraints and view rendered schedules.

Use Case Number:	UC-02
Use Case Name:	Employee Registration
Overview:	A page to register employees (nurses, physicians, etc.) who work at the hospital along with their specializations, personal hard/soft constraints, etc.
Actors:	User
Pre condition:	Users should be registered.
Flow:	Once a user is on the employee registration page, they can add/delete/modify data for each employee. This data is then fed to the backend.

Use Case Number:	UC-03
Use Case Name:	Admin privileges
Overview:	Admin will have to login first and after that he/she can view the pending leave requests of employees. Along with this, admin will get notifications about any changes in the schedule by the Optaplanner.
Actors:	Admin
Pre condition:	Admin should be logged in.
Flow:	Admin can accept/reject the requests.

Use Case Number:	UC-04
Use Case Name:	Default schedule page
Overview:	This page shows the current schedule of the employee, including the shifts and their timings.
Actors:	User
Pre condition:	Users should be registered, employees' data and default schedule should already be fed.
Flow:	Users check their schedule.

Use Case Number:	UC-05
Use Case Name:	Leave requests.
Overview:	Users can apply for leave requests from this page. Users have to select date, timing and enter a reason for leave, which will be sent to the admin for approval.
Actors:	Users
Pre condition:	Users should be registered, employees' data and default schedule should already be fed.
Flow:	User applies for a leave request.

Use Case Number:	UC-06
Use Case Name:	Providing the constraints to OptaPlanner
Overview:	OptaPlanner works behind the scenes by fusing powerful Artificial Intelligence optimization algorithms with highly effective score calculation and other cutting-edge constraint resolution methods.
Actors:	Backend
Pre condition:	Employees' data, default schedule and hard/soft constraints should be updated.
Flow:	Processes all the collected data and sends it to Optaplanner.

Use Case Number:	UC-07
Use Case Name:	Rendering generated schedule
Overview:	The schedule obtained as a result of running OptaPlanner is updated on a webpage on our website.
Actors:	Frontend
Pre condition:	User should have uploaded all required data.
Flow:	The default schedule will be updated to match entered constraints.

Use Case Number:	UC-08
Use Case Name:	Adding dynamic changes
Overview:	Constraints can be modified in order to dynamically change the schedule.
Actors:	Backend
Pre condition:	None
Flow:	Optaplanner will generate a new schedule based on the new constraints added.

Use Case Number:	UC-09
Use Case Name:	Personalized notifications and google calendar integration
Overview:	The user could check their schedule on google calendar

Actors:	Google API
Pre condition:	Google account of the user should be integrated with the framework
Flow:	The changes will get updated dynamically