Product Design

Team 39
Exam Timetable Generator

Ahana Talukdar (*2023115013)*
Raghav Grover (*2023101102)*
Keerthana Korlapati (*2023101121)*
Ananya Kasavajhala (*2023113025)*
Himani Das (*2023113020)*

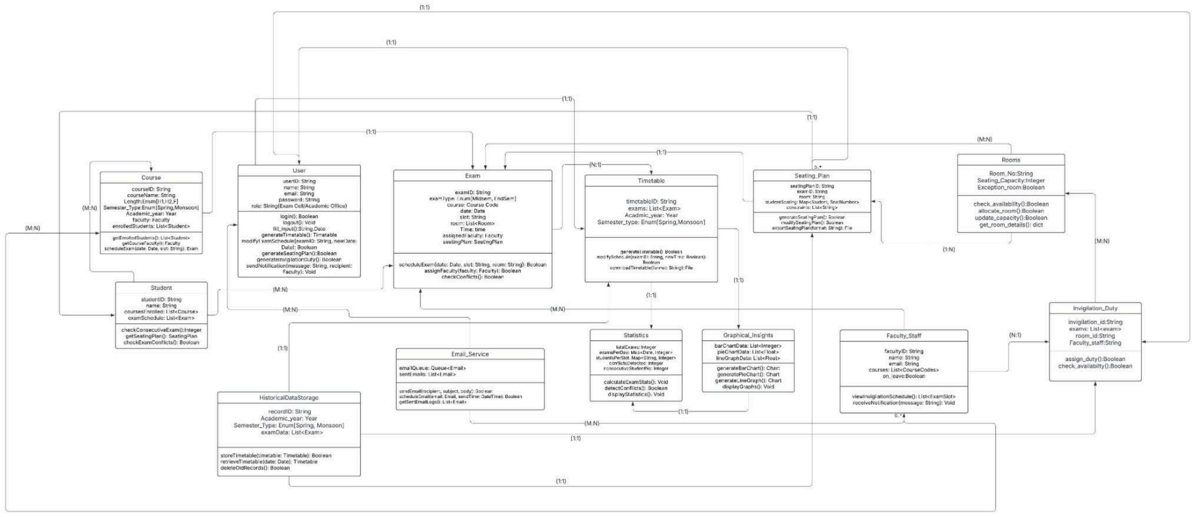
Problem Statement

The task is to automate the process of generating optimized examination timetables for Mid-Semester and End-Semester exams for the Examination Cell of IIITH. The solution involves designing algorithms that extract data such as course codes, student enrollments, staff and faculty names and courses taken by them and examination rooms details from a database to create a timetable that minimizes the number of students with consecutive exam slots in a day, seating plan for students, invigilation duties for each room,and provides statistical insights. The generated timetable should be available as both Excel and PDF documents.

Design Model

https://lucid.app/lucidchart/220429ba-fd39-486d-8b22-564bdae7cbf9/edit?viewport_loc=-1549%2C-1002%2C9006%2C3800%2CHWEp-vi-RSFO&invitationId=inv_57b54b35-62c5-49e1-b437-2429994f16f0

Image is given below


Course
courseID: String
courseName: String
Length:Enum[H1,H2,F]
Semester_Type:Enum[Spring,Monsoon]
Academic_year: Year
faculty: Faculty
enrolledStudents: List<Student>
getEnrolledStudents(): List<Student>
getCourseFaculty(): Faculty
scheduleExam(date: Date, slot: String): Exam
User
userID: String
name: String
email: String
password: String
role: String(Exam Cell/Academic Office)
login(): Boolean
logout(): Void
fill_input():String,Date
generateTimetable(): Timetable
modifyExamSchedule(examID: String, newDate: Date): Boolean
generateSeatingPlan():Boolean
generateInvigilationDuty(): Boolean
sendNotification(message: String, recipient: Faculty): Void
Exam
examID: String
examType: Enum[Midsem, EndSem]
course: Course Code
date: Date
slot: String
room: List<Room>
Time: time
assignedFaculty: Faculty
seatingPlan: SeatingPlan
scheduleExam(date: Date, slot: String, room: String): Boolean
assignFaculty(faculty: Faculty): Boolean
checkConflicts(): Boolean
Timetable
timetableID: String
exams: List<Exam>
Acadmic_year: Year
Semester_type: Enum[Spring,Monsoon]
generateTimetable(): Boolean
modifySchedule(examID: String, newTime: Boolean): Boolean
downloadTimetable(format: String): File
Seating_Plan
seatingPlanID: String
examID: String
room: String
studentSeating: Map<Student, SeatNumber>
constraints: List<String>
generateSeatingPlan(): Boolean
modifySeatingPlan(): Boolean
exportSeatingPlan(format: String): File
Rooms
Room_No:String
Seating_Capacity:Integer
Exception_room:Boolean
check_availability():Boolean
allocate_room():Boolean
update_capacity():Boolean
get_room_details(): dict
Student
studentID: String
name: String
coursesEnrolled: List<Course>
examSchedule: List<Exam>
checkConsecutiveExam():Integer
getSeatingPlan(): SeatingPlan
checkExamConflicts(): Boolean
Email_Service
emailQueue: Queue<Email>
sentEmails: List<Email>
sendEmail(recipient, subject, body): Boolean
scheduleEmail(email: Email, sendTime: DateTime): Boolean
getSentEmailLogs(): List<Email>
Statistics
totalExams: Integer
examsPerDay: Map<Date, Integer>
studentsPerSlot: Map<String, Integer>
conflictsDetected: Integer
consecutiveStudentNo: Integer
calculateExamStats(): Void
detectConflicts(): Boolean
displayStatistics(): Void
Graphical_Insights
barChartData: List<Integer>
pieChartData: List<Float>
lineGraphData: List<Float>
generateBarChart(): Chart
generatePieChart(): Chart
generateLineGraph(): Chart
displayGraphs(): Void
Faculty_Staff
facultyID: String
name: String
email: String
courses: List<CourseCodes>
on_leave:Boolean
viewInvigilationSchedule(): List<ExamSlot>
receiveNotification(message: String): Void
Invigilation_Duty
invigilation_id:String
exams: List<exam>
room_id:String
Faculty_staff:String
assign_duty():Boolean
check_availabilty():Boolean
HistoricalDataStorage
recordID: String
Academic_year: Year
Semester_Type: Enum[Spring, Monsoon]
examData: List<Exam>
storeTimetable(timetable: Timetable): Boolean
retrieveTimetable(date: Date): Timetable
deleteOldRecords(): Boolean
(1:1)
(M:N)
(N:1)
(1:N)
0..*

User	Class state ● It holds the User’s email Id, Name, Password and role. In our project only allowed Users are the Exam Cell and Academic Office. Class behavior ● It implements authentication functionalities like login, logout. Other functions are to provide input details, generate timetable, modify timetable, seating plan, invigilation duty and send faculty and staff on duty an email notification.
Student	Class state ● It holds the Student details like StudentID(Roll No), Name, Courses Enrolled in and their exam schedule. Class behavior ● It implements functions to check if there is any clashing between exams, if there are any consecutive exams and their seating plan.
Courses	Class state ● It holds the course details like CourseID(Course Code), Course Name, in which academic year and semester it is being taught, the faculty teaching and the roll nos of the students enrolled. Class behavior ● It implements functions to get the list of students enrolled under a course, the faculty involved, and get the exam schedule for that course.
Faculty_Staff	Class state ● It holds the faculty or staff ID, their name, email address, courses they teach and whether they are on leave or not. Class behavior ● It implements functions to view their invigilation schedule and receive reminder notifications a day before their invigilation duty.
Rooms	Class state ● It holds the faculty or staff ID, their name, email address, courses they teach and whether they are on leave or not. Class behavior ● It implements functions to view their invigilation schedule and receive reminder notifications a day before their invigilation duty.
Exam	Class state ● It holds the exam details for a particular course, the exam ID, the type of exam, the course, the date, time , rooms allocated, faculty involved and the seating plan. Class behavior ● It implements functionalities like scheduling a particular exam, to assign a faculty, and check its conflicts with other exams.
Timetable	Class state ● It holds the timetable details like the list of exams, the academic year and the semester type it is for. Class behavior

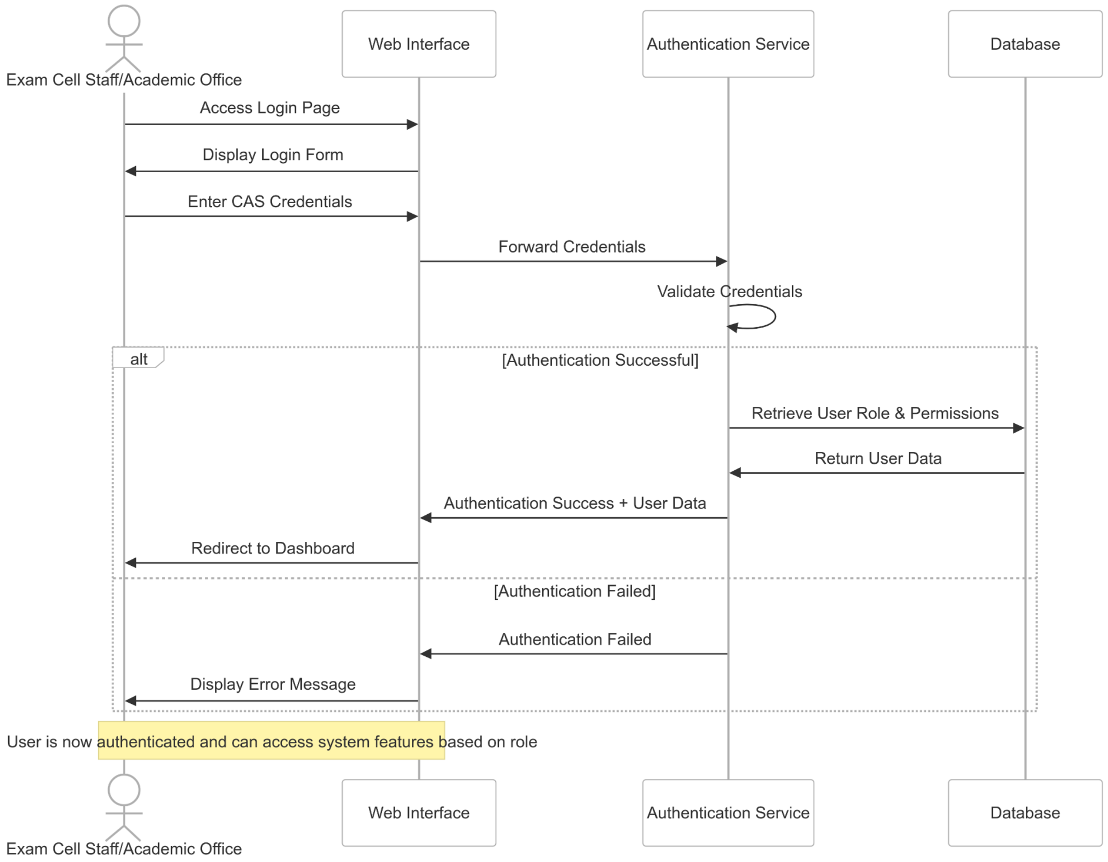
	● It implements functionalities like generating the timetable from the inputs given by the user, then the feature to modify the timetable and download it in excel or pdf format
Invigilation Duty	Class state ● It holds the invigilation duty details like the for every exam and room who are the faculty and staff who will be invigilating. Class behavior ● It implements functionalities like assigning duties to faculties and staff who are not on leave by checking their availability.
Seating_Plan	Class state ● It holds details like for a particular exam schedule the mapping between student and their seat number in a room. Class behavior ● It implements functionalities like generating the seating plan, modifying it and can be downloaded.
Historical Data Storage	Class state ● It holds details like for a particular academic year, semester type and the exam timetable. Class behavior ● It implements functionalities like storing the timetable, retrieval and deletion.
Email_Service	Class state ● It holds details like the email details and the email ids. Class behavior ● It implements functionalities like sending the email on the specified time.
Statistics	Class state ● It holds details like the total exams in a day, the students in every slot, the number of unscheduled courses and the list of consecutive students Class behavior ● It implements functionalities like generating the statistics from the data, detecting if there are any conflicts and displaying them
Graphical Insights	Class state ● It holds the graphical data like the barchart, pie chart and line graph. Class behavior ● It implements functionalities of generating the different graphs.

.

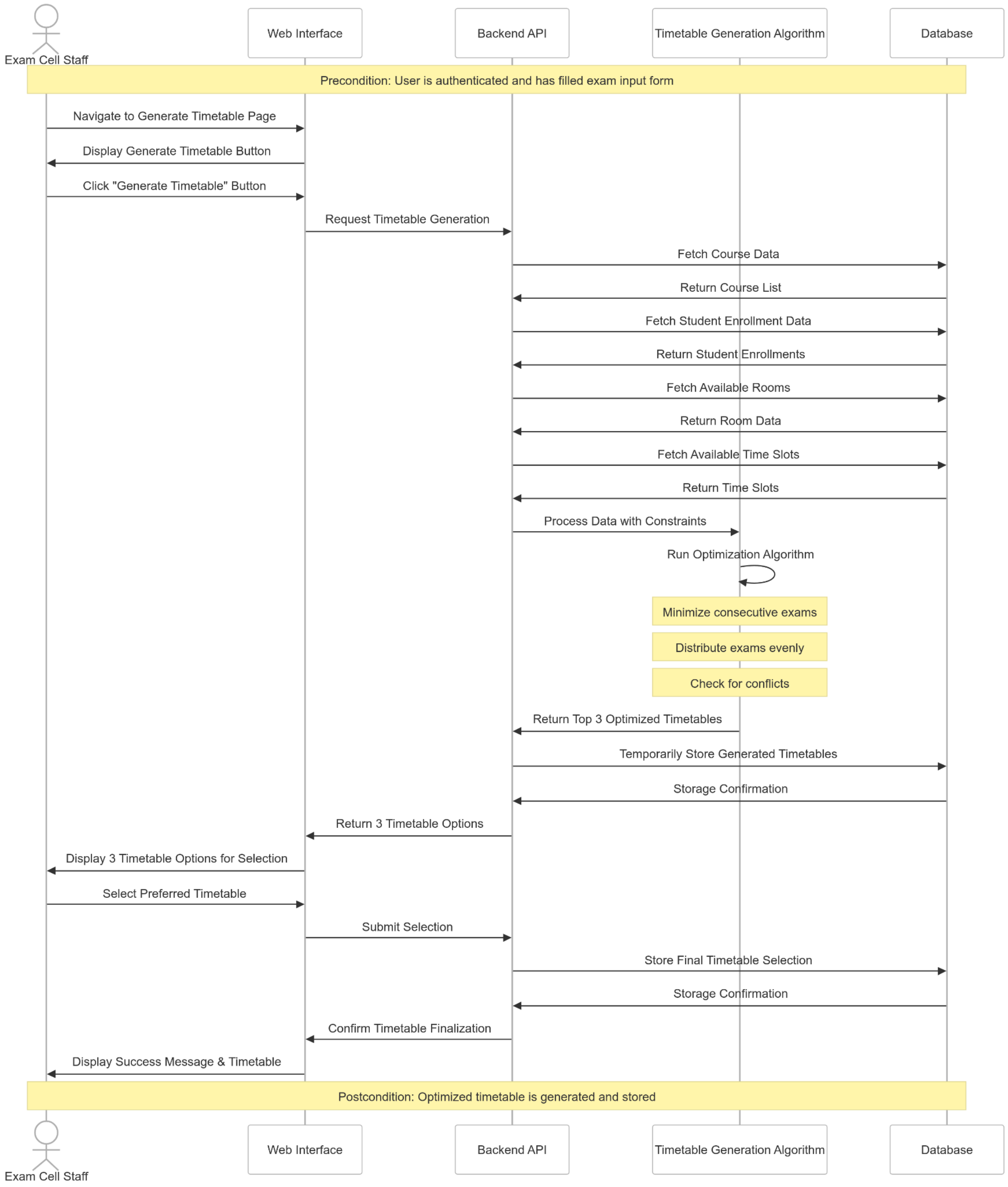
Sequence Diagram(s)

Our sequence diagrams are as follows

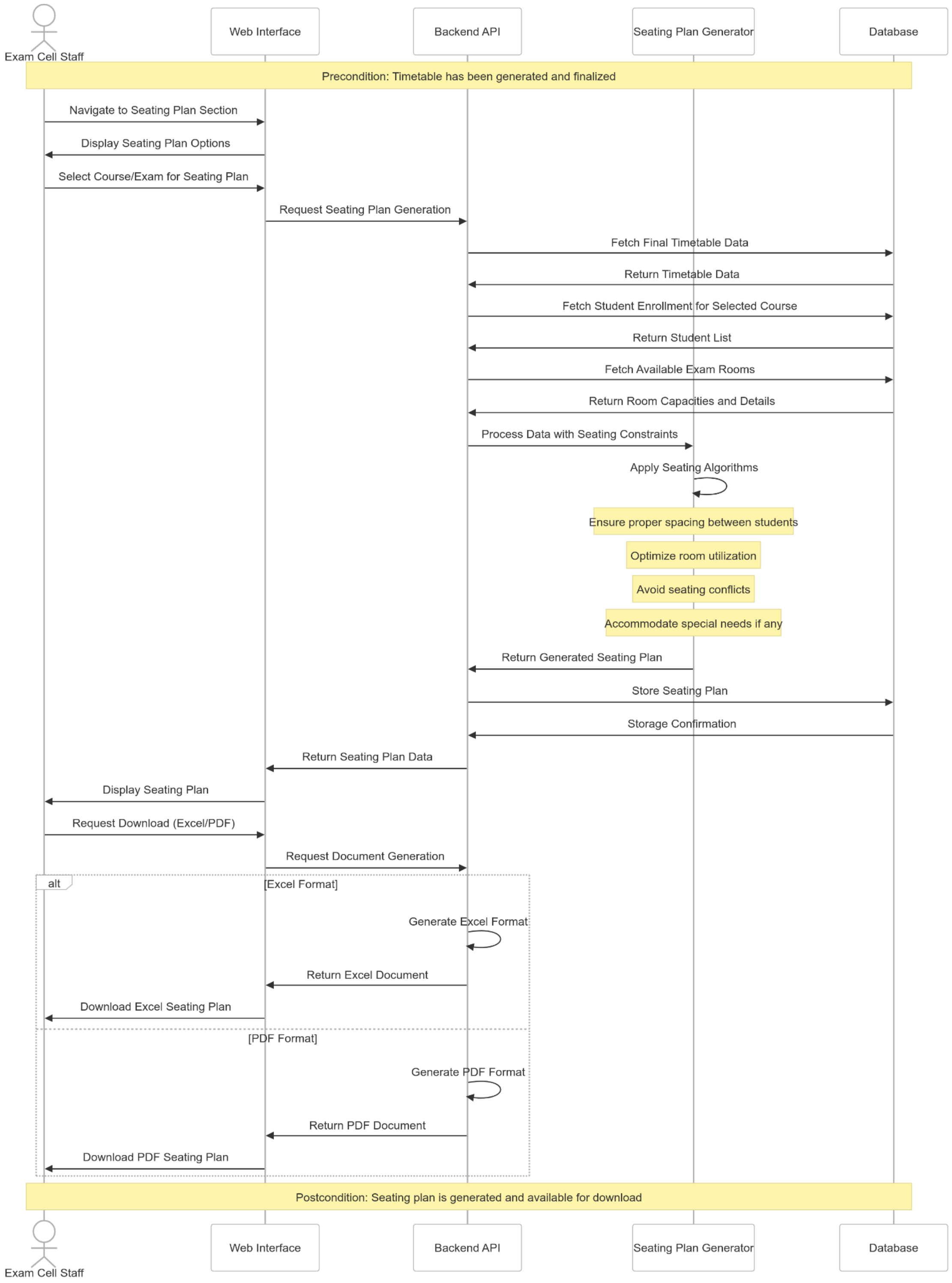
User Authentication



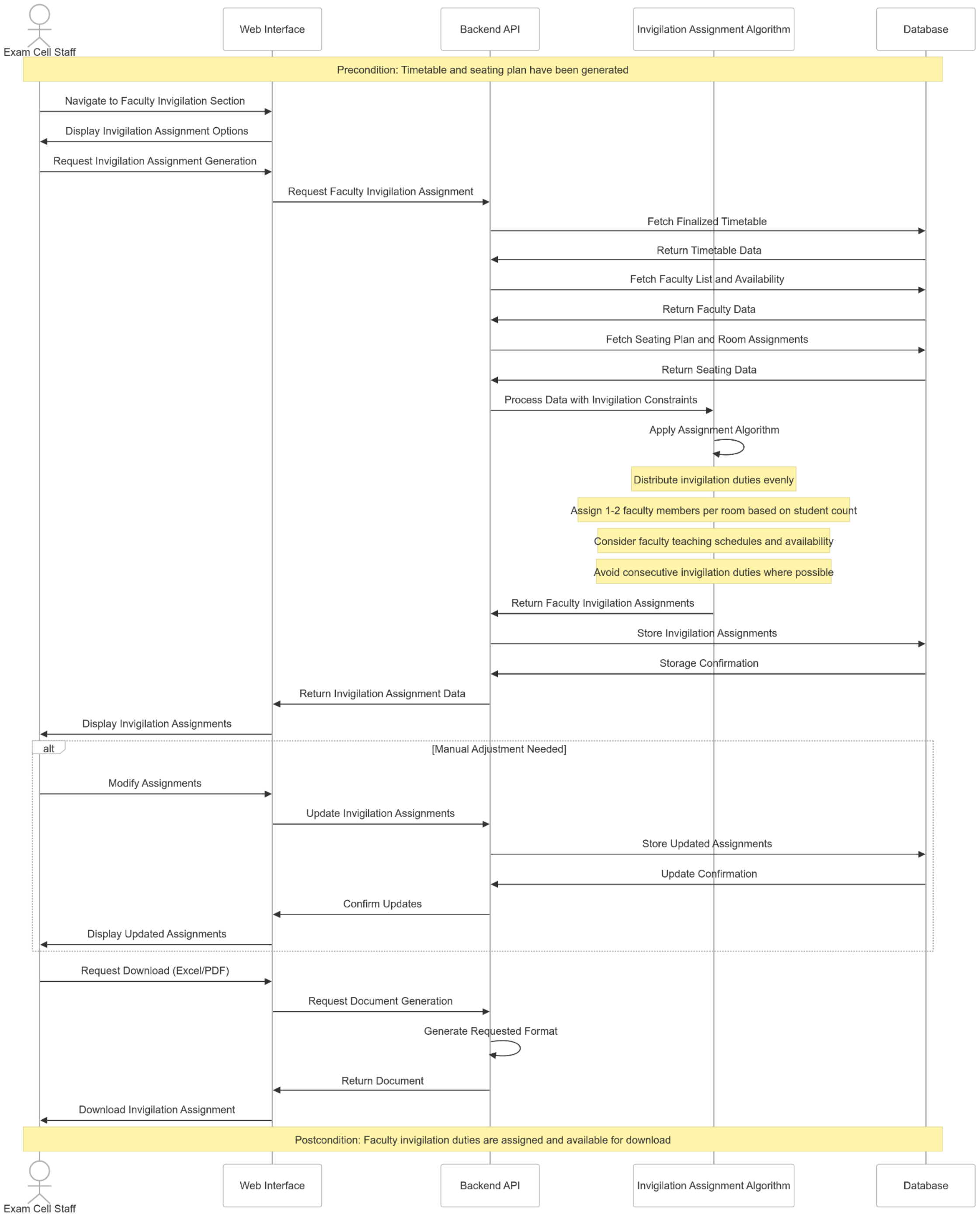
Generation of Timetable


Exam Cell Staff
Web Interface
Backend API
Timetable Generation Algorithm
Database
Precondition: User is authenticated and has filled exam input form
Navigate to Generate Timetable Page
Display Generate Timetable Button
Click "Generate Timetable" Button
Request Timetable Generation
Fetch Course Data
Return Course List
Fetch Student Enrollment Data
Return Student Enrollments
Fetch Available Rooms
Return Room Data
Fetch Available Time Slots
Return Time Slots
Process Data with Constraints
Run Optimization Algorithm
Minimize consecutive exams
Distribute exams evenly
Check for conflicts
Return Top 3 Optimized Timetables
Temporarily Store Generated Timetables
Storage Confirmation
Return 3 Timetable Options
Display 3 Timetable Options for Selection
Select Preferred Timetable
Submit Selection
Store Final Timetable Selection
Storage Confirmation
Confirm Timetable Finalization
Display Success Message & Timetable
Postcondition: Optimized timetable is generated and stored
Exam Cell Staff
Web Interface
Backend API
Timetable Generation Algorithm
Database

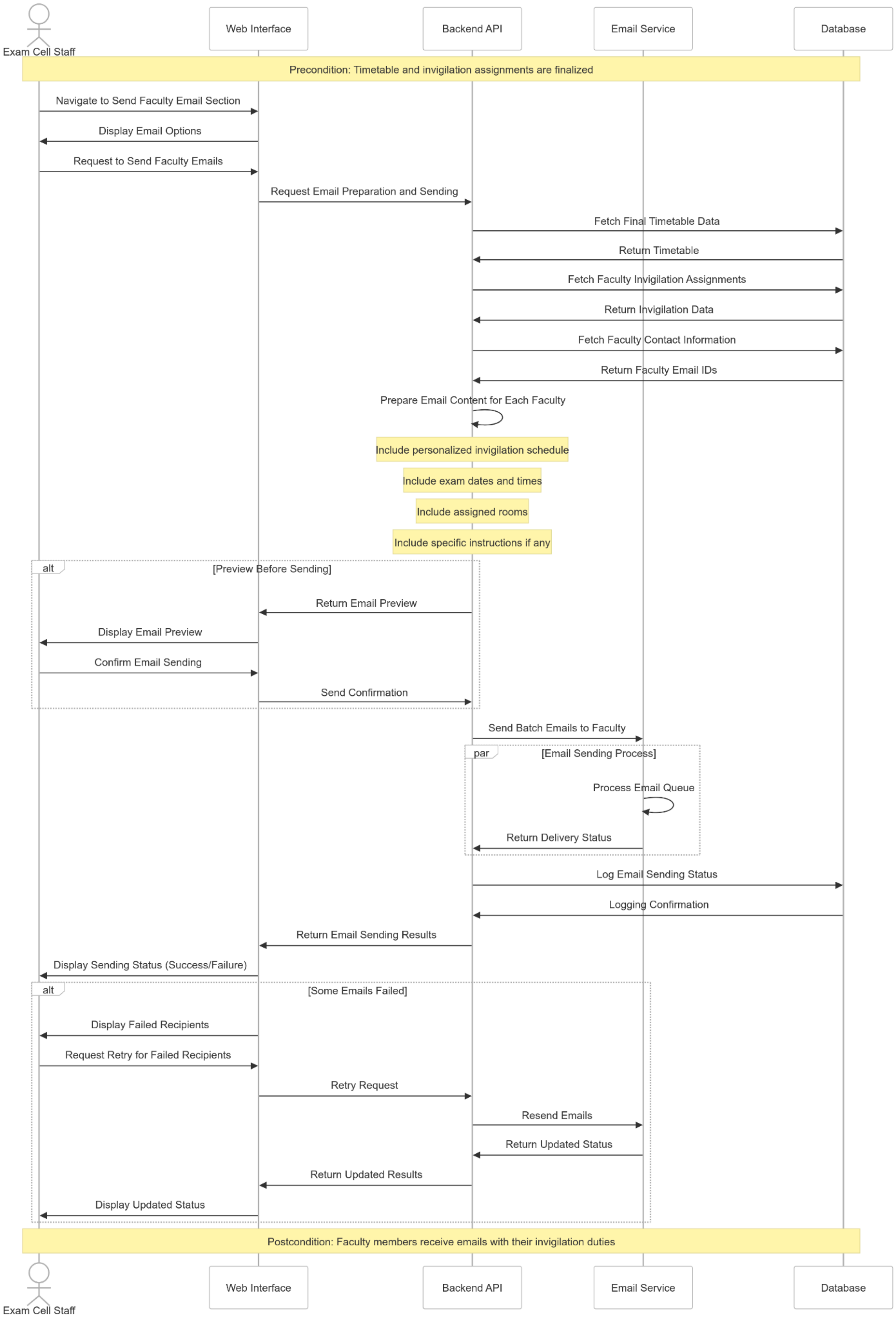
Generation Of Seating Plan


Exam Cell Staff
Web Interface
Backend API
Seating Plan Generator
Database
Precondition: Timetable has been generated and finalized
Navigate to Seating Plan Section
Display Seating Plan Options
Select Course/Exam for Seating Plan
Request Seating Plan Generation
Fetch Final Timetable Data
Return Timetable Data
Fetch Student Enrollment for Selected Course
Return Student List
Fetch Available Exam Rooms
Return Room Capacities and Details
Process Data with Seating Constraints
Apply Seating Algorithms
Ensure proper spacing between students
Optimize room utilization
Avoid seating conflicts
Accommodate special needs if any
Return Generated Seating Plan
Store Seating Plan
Storage Confirmation
Return Seating Plan Data
Display Seating Plan
Request Download (Excel/PDF)
Request Document Generation
alt
[Excel Format]
Generate Excel Format
Return Excel Document
Download Excel Seating Plan
[PDF Format]
Generate PDF Format
Return PDF Document
Download PDF Seating Plan
Postcondition: Seating plan is generated and available for download
Exam Cell Staff
Web Interface
Backend API
Seating Plan Generator
Database

Generation of Invigilation


Exam Cell Staff
Web Interface
Backend API
Invigilation Assignment Algorithm
Database
Precondition: Timetable and seating plan have been generated
Navigate to Faculty Invigilation Section
Display Invigilation Assignment Options
Request Invigilation Assignment Generation
Request Faculty Invigilation Assignment
Fetch Finalized Timetable
Return Timetable Data
Fetch Faculty List and Availability
Return Faculty Data
Fetch Seating Plan and Room Assignments
Return Seating Data
Process Data with Invigilation Constraints
Apply Assignment Algorithm
Distribute invigilation duties evenly
Assign 1-2 faculty members per room based on student count
Consider faculty teaching schedules and availability
Avoid consecutive invigilation duties where possible
Return Faculty Invigilation Assignments
Store Invigilation Assignments
Storage Confirmation
Return Invigilation Assignment Data
Display Invigilation Assignments
alt
[Manual Adjustment Needed]
Modify Assignments
Update Invigilation Assignments
Store Updated Assignments
Update Confirmation
Confirm Updates
Display Updated Assignments
Request Download (Excel/PDF)
Request Document Generation
Generate Requested Format
Return Document
Download Invigilation Assignment
Postcondition: Faculty invigilation duties are assigned and available for download
Exam Cell Staff
Web Interface
Backend API
Invigilation Assignment Algorithm
Database

Send Faculty and staff emails


Exam Cell Staff
Web Interface
Backend API
Email Service
Database
Precondition: Timetable and invigilation assignments are finalized
Navigate to Send Faculty Email Section
Display Email Options
Request to Send Faculty Emails
Request Email Preparation and Sending
Fetch Final Timetable Data
Return Timetable
Fetch Faculty Invigilation Assignments
Return Invigilation Data
Fetch Faculty Contact Information
Return Faculty Email IDs
Prepare Email Content for Each Faculty
Include personalized invigilation schedule
Include exam dates and times
Include assigned rooms
Include specific instructions if any
alt
[Preview Before Sending]
Return Email Preview
Display Email Preview
Confirm Email Sending
Send Confirmation
Send Batch Emails to Faculty
par
[Email Sending Process]
Process Email Queue
Return Delivery Status
Log Email Sending Status
Logging Confirmation
Return Email Sending Results
Display Sending Status (Success/Failure)
alt
[Some Emails Failed]
Display Failed Recipients
Request Retry for Failed Recipients
Retry Request
Resend Emails
Return Updated Status
Return Updated Results
Display Updated Status
Postcondition: Faculty members receive emails with their invigilation duties
Exam Cell Staff
Web Interface
Backend API
Email Service
Database

Design Rationale

Frontend Technology Selection

Decision: We chose **React** for the frontend instead of using basic HTML and CSS.

Alternatives Considered:

- **HTML, CSS, and JavaScript** – A simpler approach with minimal dependencies.
- **React** – A modern UI framework offering component-based architecture.

Rationale:

- React provides a more scalable and maintainable architecture compared to plain HTML/CSS.
- It supports dynamic rendering, making it easier to update timetables in real-time.
- React's ecosystem includes reusable components, improving code efficiency.
- The need for an interactive and user-friendly UI made React a better fit.

Timetable Generation Algorithm

Decision: Instead of using a **Graph Coloring** approach, we opted for a **Generic Timetable Generation** algorithm.

Alternatives Considered:

- **Graph Coloring Method** – Assigning exam slots as colors in a conflict graph.
- **Generic Timetable Generation** – Using a heuristic-based method that accounts for constraints dynamically.

Rationale:

- The graph coloring method, while theoretically sound, does not directly address all constraints such as invigilation duty allocation and seating plans.
- The generic approach allows more flexibility in handling edge cases like exam slot adjustments based on real-world constraints.
- It enables us to integrate additional optimization criteria like minimizing consecutive exams for students, which is harder to achieve with strict graph-based constraints.

- Also our client wanted multiple optimized timetable options to choose from so this algorithm generate different output results in different iterations
- It is easier to debug and modify according to evolving institutional needs.

Output Format Selection

Decision: The timetable will be generated in **Excel and PDF** formats.

Alternatives Considered:

- **Web-based only (display on UI)**
- **Excel and PDF**

Rationale:

- Exam coordinators prefer working with spreadsheets and printed documents.
- Excel allows further manual modifications if needed.
- PDFs provide a standardized format for sharing and printing without layout changes.
- Supporting both ensures accessibility and flexibility for different user needs.