Comprehensive Exercise Report

Team The Frenchof Section 000(didn’t find the section number)

Group Member Names and Unity IDs

- Nom Michael Yaghi – 250AEB040
- Noah N’Diaye Falcy – 250AEB039
- Clovis Le Floc’h – 250AEB050
- Maxime Viti – 250AEB014
- Axel Pineau – 250AEB022

Requirements/Analysis

The main goal of our project is to develop an interactive portfolio website for a psychologist who wants to present her services and manage her appointments online. The website is meant to be professional, informative, and functional.

Main Objectives:

1. Professional Presentation
    a. Provide clear and visually appealing pages presenting the psychologist, her background, specializations, working methods, and pricing.
    b. Make all necessary information available (contact details, practice location, working hours, etc.) to build trust with visitors.
2. User Account Management
    a. Integrate a secure registration and login system for clients.
    b. Each user will have access to a personal space to book, view, or cancel appointments.
3. Booking System
    a. The psychologist will be able to set her availability through a configurable time slot table (by day/hour).
    b. Clients can view available time slots, book an appointment online, and receive a confirmation.
    c. A management system also allows the psychologist to modify or block her availability at any time.

Target Users:

- Potential clients looking to book a session with a psychologist easily and quickly online.
- The psychologist herself, aiming to automate and simplify her appointment management.

Secondary Objectives (optional depending on available time):

- Add an email notification system.
- Include a secure contact form.
- Potential integration of video consultations.

Journal (W2)

Project description in one sentence:

A professional website allowing a psychologist to showcase her services and manage appointments via an online booking system.

Known client requirements:

- Presentation, services, pricing, and contact pages
- Registration and login system
- Appointment time slot management
- User-friendly and responsive interface
- Adherence to a design brief (fonts, color palette)

Questions and answers from the client:

- Q: Do you want to offer video consultations directly through the website?
  A: Not for now, maybe later.
- Q: What kind of confirmation for appointments?
  A: An email confirmation will suffice.

Unfamiliar topics / Research:

- Setting up a dynamic calendar system
- Integrating a MongoDB database (https://www.mongodb.com/docs/)

User descriptions:

- Adult patients who want to book a consultation
- The psychologist who manages availability and views bookings

User interaction:

- The patient creates an account, views time slots, and books a session
- The psychologist creates/updates availability and views reservations

Essential features:

- Registration and login
- Smooth navigation between site pages
- Booking a time slot
- Appointment confirmation and management

Other notes:

- Focus on ease of use for both user types
- Interface must be compatible with mobile and desktop

Software Requirements

Functional Requirements

1. Authentication
    a. Clients can create an account.
    b. Clients can log in using an email and password.
    c. Clients can log out.

2. Navigation
    a. Users can access pages: Home, About, Services, Pricing, Contact.
    b. The site must be accessible via web browsers on desktop and mobile.
3. Booking
    a. Clients can view available time slots defined by the psychologist.
    b. Clients can book an appointment.
    c. The psychologist can define, update, and delete her availability.
4. Confirmation
    a. An email confirmation is sent after booking.
5. User Dashboard
    a. Each user has a personalized dashboard.
    b. Clients can view and cancel their appointments.
    c. The psychologist can view upcoming reservations.

Non-Functional Requirements

1. Security
    a. User data must be stored securely (e.g., hashed passwords).
    b. Role-based access (psychologist vs. client).
2. Accessibility
    a. The site must be responsive (adapted to phone, tablet, and desktop screens).
3. Performance
    a. Pages should load in under 2 seconds.
4. Maintainability
    a. Code must be structured for easy future additions/updates (modular architecture).
5. Compliance
    a. Comply with GDPR standards for personal data handling.

Black-Box Testing

Input Types:

- Registration form: first name, last name, email, phone number, password.
- Login form: email, password.
- Appointment booking selection by the client.
- Availability definition/modification by the psychologist.

Output Types:

- Confirmation messages for registration, login, and booking.
- Display of available/reserved time slots.

- Error messages for invalid input or unauthorized access.

Equivalence Classes:

- Registered vs. unregistered user.
- Valid vs. invalid credentials.
- Available vs. unavailable time slot.
- Successful booking vs. attempt without login.

Boundary Values:

- Time slots: start at 10:00 AM, end at 7:00 PM.
- Minimum password length (to be specified, e.g., 8 characters).
- Max field lengths (e.g., name, message).

Special Cases to Test:

- Attempting to access booking pages without login.
- Booking a time slot outside of allowed hours.
- Double submission of forms (spam protection).

Other Notes:

- All tests must be independent of internal code structure (true black-box).
- The UI must clearly guide the user in case of error (e.g., required field message).

Journal (W4)

During week 4, after finalizing the website mockup in collaboration with the psychologist, we began setting up the technical foundation of the project. We created a GitHub repository to centralize the code and facilitate collaboration among team members.

We then installed and configured the project's core technologies:

- React for the frontend
- Vite as the development environment
- MongoDB for the database

We planned the project architecture and file organization (components, services, routes, etc.). A quick initial test was conducted to ensure everything was set up correctly and that development tools were working.

This step laid a solid foundation for further development while clarifying technical task distribution among the group members.

Black-box Test Cases

Test ID	Description	Expected Results	Actual Results
TC01	User logs in with a valid email and password	Redirected to their personal dashboard	done
TC02	User enters an invalid email format	Error message: "Invalid email format"	done
TC03	Unauthenticated user tries to access booking page	Redirected to login page	done
TC04	User registers with all valid input fields	Account is created and redirected to the main page	done
TC05	User registers with missing required field (e.g., email)	Error message is displayed	done

TC06	User books a valid time slot	Booking is confirmed and shown in dashboard	done
TC08	Psychologist edits her availability successfully	Availability is updated in the system	
TC10	User tries to access dashboard without login	Redirected to login page	done
TC11	Psychologist creates overlapping availability slots	Error message: "Time slot already exists"	done
TC12	Client cancels a previously booked appointment	Booking is removed and slot becomes available	done
TC13	User attempts login with an unregistered email	Error: "Account not found"	done

Software Design

Our software design is directly informed by the project's implementation and reflects a full-stack application using React, Node.js, Express, and MongoDB.

Backend (Node.js/Express)

Models

- User
  Attributes: nom, prenom, email, password, telephone, role
  Passwords are hashed using bcrypt. Each user has a role: client or psychologue.
- Slot
  Attributes: date, estDisponible, prisPar, psy, dateAnnulation
  Each slot is linked to a psychologist and can be reserved or canceled.

Controllers

- authController.js: Handles registration and login logic with token generation.

- slotController.js: Manages slot generation, availability, booking, cancellation, update, and deletion.
- Middleware: Includes JWT-based authentication and role-based access control (authorizeRole).

Routes

- /auth: login and registration
- /slots: create, update, delete, reserve, and retrieve slots
- Secured by middleware (authentication + role)

Frontend (React)

- Components: Login, Register, Dashboard, Booking, AvailabilityForm
- Routing: handled by react-router-dom
- Axios: used to call backend APIs
- Auth: JWT token stored client-side, used to secure routes

Data Flow

1. A user registers or logs in, receiving a token.
2. The frontend stores the token and uses it in headers for secured routes.
3. The backend verifies the token and grants access to protected operations.
4. MongoDB stores all data (users, slots, reservations).
5. Emails are sent for confirmation and cancellation notifications.

Journal (W6)

During week 6, we progressed significantly in setting up the backend and frontend structure of the project based on the finalized design. We began coding key modules including user authentication (register/login), and initial API routes for managing user data and availability slots.

The backend (Node.js/Express) was connected to the MongoDB database, with schema definitions for users and time slots. Basic CRUD operations were tested using tools like Postman to validate API responses.

On the frontend (React), we created the basic routing structure using react-router-dom, and started building the interface components: the login and registration pages were designed and linked to the backend.

Overall, this week was focused on laying the foundations for user management and preparing the booking logic, ensuring that the system components can communicate properly via RESTful endpoints.

Implementation

The implementation was structured around a full-stack JavaScript environment using React for the frontend and Node.js/Express with MongoDB for the backend. Below are the main highlights of how we carried out the implementation phase:

Technologies Used

Frontend: React, React Router DOM, Axios

- Backend: Node.js, Express, Mongoose
- Database: MongoDB Atlas
- Authentication: JSON Web Token (JWT), bcrypt for password hashing
- Email Service: Custom utility integrated with appointment logic
- Key Implementation Aspects

User registration and login: Secure account creation, role management (client vs. psychologist), password hashing, and JWT token generation.

- Protected routes: Middleware validates JWT tokens and checks user roles for access control.
- Slot management: Psychologists can create, edit, delete, and view time slots.
- Booking system: Clients can book and cancel available time slots. The system prevents bookings for already reserved or expired slots.
- Database models: Mongoose schemas for User and Slot ensure data integrity.
- Email notifications: Automated emails are sent for appointment confirmations and cancellations.
- Challenges Faced

Synchronizing frontend and backend logic for slot availability.

- Validating overlapping and expired time slots.
- Securing API endpoints with robust authentication and authorization checks.
-

Journal (W8)

During week 8, we focused on implementing key user functionalities and solidifying communication between the frontend and backend.

On the backend, we completed the development of secure JWT-based authentication, role-based access control, and slot-related APIs. This included the ability to create, update, delete, and reserve time slots, while also enforcing restrictions like avoiding duplicate or expired bookings.

On the frontend, we integrated login and registration forms with the backend API and implemented protected routes using React Router. Axios was configured globally to include JWT tokens in requests. We also began integrating appointment booking components, making real-time API calls to check slot availability and post reservations.

We performed several integration tests to ensure seamless interaction between both ends and validated major flows like login → booking → dashboard update.

Implementation Details

The application is designed to streamline psychologist appointment management while providing a smooth user experience.

User Authentication

- Uses JWT for secure session handling.
- Passwords are hashed with bcrypt before storage.
- Users can register with their role (client or psychologist) and log in to access protected features.

Slot Management

- Psychologists generate slots by selecting dates and configuring time ranges for each weekday.
- Slots are stored in the database with timestamps and availability flags.
- Validation prevents double-booking or scheduling in the past.

Booking Workflow

- Clients browse available slots and submit a booking.
- The slot becomes unavailable immediately upon reservation.
- Both users and psychologists can view their slots/reservations.
- Clients can cancel only if more than 24h remain before the appointment.

Email Integration

- Confirmations and cancellations trigger email notifications using a custom utility and template.

Interface

- The frontend is built with responsive components.
- Uses Axios to communicate with the backend.

- React Router ensures navigation across components like Login, Register, Dashboard, Booking.

Admin Protection

- Routes like availability management and deletion are restricted to psychologists via role-based middleware.

Testing

To validate the system, we conducted manual and API-based testing focused on core features like authentication, slot management, and booking logic.

Functional Testing

- Login/Register: Tested valid and invalid credentials, duplicate emails, and empty fields.
- Protected Routes: Verified access is blocked without valid JWT, and role-based access works correctly.
- Slot Management: Created, updated, and deleted slots, ensuring rules were enforced (e.g., no past slots).
- Booking Logic: Confirmed clients cannot book unavailable or expired slots, and cancellation works if >24h before.

Tools Used

- Postman: For manual API testing.
- Console Logs: For backend behavior during development.
- Browser Testing: Chrome, Firefox for frontend validation.

Observations

- All core flows performed as expected.
- Error handling proved robust, with clear messages returned to frontend.
- Authentication and role-based protections were effective across endpoints.

Journal (W10)

During week 10, we dedicated time to thorough testing and bug fixing across all major features. We simulated various user flows including account creation, login,

booking, and cancellation. Edge cases such as expired tokens, invalid inputs, or double booking were specifically targeted.

We validated API security using Postman and ensured that role-based access was consistently enforced across endpoints. Email notifications were tested for reliability and content accuracy.

Final adjustments were made to improve UI clarity and error feedback on the frontend. With testing results documented and code cleaned up, we prepared the project for final presentation and delivery.

Presentation – Instructions: Week 12

Preparation

Give a brief description of your final project
We developed a full-stack web application that allows a psychologist to manage online appointments. The app enables secure user registration, availability management, appointment booking, and email notifications.

Describe your requirement assumptions/additions.
We assumed psychologists would use weekly time slot templates.
We added a 24-hour cancellation restriction and implemented email notifications for bookings and cancellations.

Describe your design options and decision. How did you weigh the pros and cons of the different designs to make your decision?
We compared a monolithic architecture with a modular REST architecture. We chose the modular REST approach for its scalability, clarity, and separation of concerns. This made the system easier to maintain and extend.

How did the extension affect your design?
Adding real-time email notifications introduced asynchronous logic. We isolated this functionality into a dedicated utility module to keep controllers clean and focused.

Describe your tests (e.g., what you tested, equivalence classes).
We tested:

- Valid and invalid authentication flows
- Route protection via JWT and role-based middleware
- Creation, modification, and deletion of availability slots
- Booking scenarios including already booked, expired, and less-than-24h cancellations
  We covered multiple equivalence classes (valid/invalid input, roles, availability states).

What lessons did you learn from the comprehensive exercise (i.e., programming concepts, software process)?

- The importance of robust error handling and validation
- How to design a modular backend architecture with Express and Mongoose
- How to implement secure authentication with JWT
- Synchronizing frontend state with backend data using React and Axios

What functionalities are you going to demo?

- User login and registration
- Psychologist managing availability slots
- Client booking an appointment
- Email confirmations and cancellations

Who is going to speak about each portion of your presentation?

- Member 1: Project overview and requirements
- Member 2: Technical design and architecture choices
- Member 3: Live demo, testing details, and conclusion
  Each member will speak for at least 2 minutes.

Other notes:

- The team rehearsed the presentation several times.
- In case of technical issues, screenshots and logs are prepared as a backup.
- Slides include diagrams for system flow, database schema, and tested use cases.