**1. Overview**

The TWAAOS-SIC system is a modular, web-based platform for managing and scheduling exams and colloquia at the Faculty of Electrical Engineering and Computer Science (FIESC), “Ștefan cel Mare” University of Suceava.  
The architecture is designed for scalability, clarity, and integration with external academic systems.

**2. Application Architecture Diagram**

┌──────────────────┐

│ Front-End UI │ ← React.js

└──────│──────────┘

│ REST API / JSON

┌──────▼────────────┐

│ API Gateway │ ← FastAPI (Python)

│ (Authentication, │

│ Routing, Rate │

│ Limiting) │

└──────│────────────┘

│

┌──────▼────────────┐

│ Exam Service │ ← Scheduling, validation, business logic

└──────│────────────┘

│

┌──────▼────────────┐

│ User Service │ ← Manages users (SG, SEC, CD, ADM)

└──────│────────────┘

│

┌──────▼────────────┐

│ Notification Svc │ ← Email + alerts (e.g., Sendgrid)

└──────│────────────┘

│

┌──────▼────────────┐

│ Orar Adapter │ ← Integration with USV Timetable APIs

└───────────────────┘

**3. Technology Stack**

| **Layer** | **Technology** | **Notes** |
| --- | --- | --- |
| Front-End | React.js | Modern SPA, user roles and workflows |
| Back-End | FastAPI (Python 3.x) | REST API, validation, async, modular |
| Database | PostgreSQL (preferred) | Relational, robust transactions |
| Local Dev | SQLite (optional) | For local development/testing only |
| Email | Sendgrid (or compatible) | Asynchronous notification service |
| Container | Docker | Dockerized for consistent deploys |
| Orar API | <https://orar.usv.ro> | Integration for schedule validation |

**4. Project Directory Structure**

project/

├── backend/

│ ├── app/

│ │ ├── main.py

│ │ ├── api/

│ │ ├── core/

│ │ ├── models/

│ │ ├── services/

│ │ └── database/

├── frontend/

│ └── src/

├── docker/

│ ├── Dockerfile.backend

│ ├── Dockerfile.frontend

│ └── docker-compose.yml

**5. User Authentication**

| **Role** | **Authentication Method** | **Mechanism/Notes** |
| --- | --- | --- |
| Secretariat | Google OAuth2 | Email domain whitelist: @usv.ro / @student.usv.ro |
| Group Leader | Google OAuth2 | Same as above |
| Professor | Google OAuth2 | Same as above |
| Administrator | Username + Password (JWT) | No Google login allowed, JWT token only |

**6. Functional Roles and Responsibilities**

* **Secretariat (SEC):** Uploads templates, notifies, edits, exports plans.
* **Group Leader (SG):** Proposes exams, sees own group only, interacts per-discipline.
* **Professor (CD):** Validates/rejects proposals, assigns rooms/assistants, avoids conflicts.
* **Admin (ADM):** Manages users, metadata, access; cannot use Google login.

**7. Data Import/Export**

* **Excel:** Used for importing exam, group, and discipline data (see Excel README and templates).
* **Export:** Supported formats are Excel, PDF, ICS (calendar).

**8. API Integration**

* Uses **USV Timetable API** endpoints:
  + cadre.php?json (staff)
  + sali.php?json (rooms)
  + facultati.php?json (faculties)
  + subgrupe.php?json (groups)
  + orarSPG.php?ID=... (timetable for professor/group)
* Data mapping and field validation rules are in the [Business Rules & Scenarios] and OpenAPI YAML.

**9. Database Schema**

* Primary relational database: **PostgreSQL**.
* See dedicated **.sql** file for CREATE TABLE, keys, and constraints.
* Uses names, not IDs, for imported Excel, for readability and traceability.

**10. Containerization (Docker)**

* **All components are dockerized**:
  + Dockerfile.backend (FastAPI)
  + Dockerfile.frontend (React)
  + docker-compose.yml (PostgreSQL, backend, frontend, reverse proxy optional)
* Developers/users run everything locally or in cloud with docker-compose up.
* All persistent data is mounted/stored in Docker volumes for safe data handling.

**11. AI Integration Recommendations**

* When used in an **AI-driven IDE** (e.g., Windsurf/Codeium), provide all documentation files:
  + Application Architecture (this document)
  + OpenAPI YAML contract (API definition)
  + Database schema (.sql)
  + Excel template with README and validations
  + Business Rules & Scenarios document
* **Do not attempt to relax or infer business rules**: All validation and workflow rules must be followed strictly as documented.
* Always log and explain every row-level error on import.
* For any uncertainty, reject the ambiguous operation and require explicit clarification.

**12. References to Other Documents**

* **Business Rules & Scenarios:** Detailed business logic, permissions, and workflows.
* **OpenAPI YAML:** REST endpoint specifications, schemas, security.
* **Database Schema (.sql):** Full relational structure, constraints.
* **Excel README:** Format, columns, and validation for imports.
* All documents must be kept in sync to avoid “gray areas” or logical mismatch.

**13. Module Responsibilities Summary**

| **Module** | **Responsibilities** |
| --- | --- |
| Front-End UI | User workflows, data entry, authentication, role-based navigation |
| API Gateway | Auth, routing, rate limiting, request validation |
| Exam Service | Exam scheduling, validation, status transitions, conflict checks |
| User Service | User CRUD, permissions, role checks |
| Notification Svc | Email, alerts, queue management, logging |
| Orar Adapter | Data synchronization & validation with external USV Timetable API |
| Database | Storage of all persistent data, transactional integrity |

**14. Change Log**

| **Version** | **Date** | **Author** | **Description** |
| --- | --- | --- | --- |
| 0.1 | 2024-05-28 | Sebastian / ChatGPT | Initial version, basic architecture |
| 0.2 | 2024-06-05 | Sebastian / ChatGPT | Added API mapping and Excel import/export |
| 0.3 | 2024-06-06 | Sebastian / ChatGPT | Clarified Docker, PostgreSQL, AI usage, refs |

**15. Contact & Support**

* **Project Maintainer:**  
  *Sebastian [<inserați adresa de email sau user dacă doriți]*
* **Support Channels:**
  + For any technical questions, implementation blockers, or clarifications regarding business logic, always check first the following files (in this order):
    1. **Business Rules & Scenarios**
    2. **OpenAPI Specification (YAML)**
    3. **Database Schema (SQL)**
    4. **Excel Import/Export README**
    5. **This Application Architecture document**
  + If the answer is not found or is ambiguous, document the question and contact the maintainer via email or the designated support channel (ex: GitHub Issues, Teams, Discord).
* **Feedback & Contributions:**
  + Any suggestion or change proposal should be submitted as a documented issue or merge request.
  + All architectural or business rule changes must be approved before implementation and reflected in all related documentation files.
* **Incident Reporting:**
  + All critical bugs, security issues, or data loss incidents must be reported immediately to the maintainer.
  + Use version control (e.g., Git) for tracking all changes and rollbacks.

**16. Deployment & Environment Guidelines**

* **Docker Compose:**  
  Use docker-compose.yml as the primary entry point for development and production setups.
  + Launches PostgreSQL database, backend API (FastAPI), frontend UI (React).
  + Environment variables for secrets, DB connections, and API keys are stored in .env files and referenced securely.
  + For production, configure HTTPS and set secure JWT/email credentials.
* **Database Initialization:**  
  On first run, run all migrations (Alembic for SQL if using FastAPI) and seed initial data if required.
  + Ensure database connection is healthy before starting backend API.
  + Backup and restore procedures should use PostgreSQL’s native tools (pg\_dump, pg\_restore).
* **Email/Notification Service:**
  + Configure Sendgrid (or compatible SMTP) with environment variables.
  + All notifications are sent asynchronously via a task queue (Celery/RQ or FastAPI BackgroundTasks).
  + Errors in notification delivery are logged but do not block exam scheduling.

**17. Security & Privacy Considerations**

* **Authentication:**
  + Enforce HTTPS in all production environments.
  + JWT tokens must be signed and stored securely.
  + Google OAuth2: Accept only whitelisted domains (@usv.ro, @student.usv.ro).
* **Authorization:**
  + All endpoints must check permissions as defined in the business rules.
  + Attempts to bypass roles or access unauthorized data are logged and rejected.
* **Data Privacy:**
  + User passwords (for ADM) must be hashed and salted (bcrypt recommended).
  + All sensitive user actions (imports, approvals, deletions) are audited.
* **GDPR/Privacy:**
  + Export and deletion requests must be possible for personal user data if required.
  + All exports are restricted to authorized users.

**18. Maintenance & Future-Proofing**

* **Scalability:**
  + The system is modular; microservices can be scaled independently in cloud environments (Kubernetes-ready).
  + Stateless services (backend, frontend) allow for horizontal scaling.
* **Extensibility:**
  + To add new features (e.g., statistics dashboards, API integrations), update both the OpenAPI YAML and business rules.
  + When adding new roles or validation logic, reflect changes in all related documents.
* **Upgrades:**
  + Docker images should be rebuilt after dependencies or environment changes.
  + Document all upgrades in the Change Log section.

**19. Troubleshooting & FAQ**

* **Q:** What to do if import fails with validation errors?  
  **A:** Check error logs for row-specific feedback. Correct data in Excel and retry.
* **Q:** API returns 401 Unauthorized?  
  **A:** Confirm JWT/OAuth2 token is present, valid, and user has necessary role.
* **Q:** How to reset admin password?  
  **A:** Use the CLI tool or database script to set a new password hash.
* **Q:** Can a group leader import exams for another group?  
  **A:** No. System enforces that SG can only import/view their own group’s data.

**20. Document Links & References**

* [Business Rules & Scenarios Document]
* [OpenAPI Specification YAML]
* [Database Schema SQL]
* [Excel Import/Export Templates & README]

*(Replace with actual file names and paths in your delivery archive.)*

**21. Glossary of Terms (if not already in other docs; for redundancy)**

* **SG:** Group Leader
* **CD:** Professor
* **SEC:** Secretariat
* **ADM:** Administrator
* **Exam:** Scheduled assessment
* **Discipline:** Subject/Course
* **Program:** Specialization
* **Faculty:** Academic division (e.g., FIESC)

**22. Document History**

Add new entries as you update:

| **Version** | **Date** | **Author** | **Description** |
| --- | --- | --- | --- |
| 0.4 | 2024-06-06 | Sebastian / ChatGPT | Added deployment, security, FAQ, links |

**23. Final Notes**

* All development, code review, and AI generation should reference this document set for every architectural or functional question.
* If any conflict exists between documents, follow this priority:
  1. **Business Rules & Scenarios**
  2. **OpenAPI YAML**
  3. **Database Schema**
  4. **Architecture Overview (this document)**
  5. **Excel Templates/README**
* If in doubt, escalate to the project maintainer for clarification **before** making assumptions.