TED UNIVERSITY

Faculty of Engineering

Computer Engineering Department

CMPE_491 Senior Design Project I

HIGH-LEVEL DESIGN REPORT

Name of the Project: JustiWise

Web Page URL: https://finalprojectjustiwise.github.io/CMPE_491/

Team Members:

Irmak Orhan

Ahmet Engin Büyükdığan

Aslı Algın

Ali Fuat Dündar

Supervisor: Tansel Dökeroğlu

Jury Members: Fırat Akba, Ayşe Yasemin Seydim

CONTENT

1. Introduction

The legal field faces numerous challenges stemming from inefficiencies in process management and gaps in professional training. Current legal practices heavily rely on traditional, manual methods, which result in prolonged case resolution times and limited access to justice. Moreover, law students and professionals often lack access to practical, interactive training environments, which hinders their readiness for real-world scenarios.

JustiWise is a cutting-edge solution designed to address these challenges through automation, innovation, and advanced educational tools. By combining AI-driven technologies and user-centric design, the platform seeks to streamline legal workflows, empower legal professionals, and enhance accessibility to legal services.

This document outlines the high-level design of the JustiWise system, focusing on its architecture, design principles, and key functionalities.

1.1 Purpose of the System

JustiWise aims to revolutionize the legal sector by leveraging artificial intelligence to address inefficiencies and gaps in professional education. The primary objectives of the system are as follows:

1. **Streamlining Legal Processes**: By automating repetitive and time-consuming tasks, such as document preparation and case analysis, JustiWise enables lawyers to focus on complex legal challenges.
2. **Enhancing Accessibility**: The platform's AI-driven consultation services provide individuals with affordable and efficient legal advice, reducing barriers to accessing justice.
3. **Improving Legal Education**: Through immersive training modules, including virtual trials and mediation simulations, JustiWise equips law students and professionals with practical experience and skills.
4. **Ensuring Compliance and Ethics**: The system adheres to strict data protection laws (KVKK and GDPR) and promotes ethical AI usage, ensuring fairness, transparency, and non-discrimination.

The integration of these functionalities into a single platform transforms JustiWise into a comprehensive tool for modernizing legal practices and advancing professional development.

1.2 Design Goals

The design of JustiWise emphasizes functionality, scalability, and compliance. The following goals have been established to guide the system's development:

1.2.1 Extensibility and Modularity

JustiWise is built to support future enhancements and integrations. Each component operates independently, allowing for the seamless addition of new features, such as expanded training scenarios or AI capabilities, without disrupting existing modules.

1.2.2 Maintainability

The system's modular structure ensures that updates or troubleshooting in one component do not impact others. Clear documentation and robust testing protocols further enhance maintainability.

1.2.3 Usability

A key priority is delivering an intuitive and user-friendly experience. The platform’s interface is designed for all user roles (clients, lawyers, students), ensuring ease of navigation and accessibility to core functionalities.

1.2.4 Performance

High performance is critical for the platform’s success. Client inquiries, document processing, and training modules are optimized for rapid response times, ensuring efficient operations.

1.2.5 Scalability

The architecture supports concurrent usage by thousands of users, accommodating growth in demand without degradation in performance.

1.2.6 Security

Advanced security measures, including encryption, two-factor authentication, and secure data storage, safeguard sensitive user information and ensure compliance with legal standards.

1.2.7 Compliance

The system complies with all relevant legal regulations, including KVKK and GDPR, and adheres to ethical AI guidelines to promote fairness, transparency, and accountability.

1.2.8 Reliability

The platform is designed for continuous operation, with failover mechanisms and redundancy to ensure availability and data integrity under varying conditions.

1.3 Definitions, Acronyms, and Abbreviations

- **AI**: Artificial Intelligence—Technology designed to simulate human intelligence for complex tasks such as decision-making and language processing.
- **Avatar**: A digital entity that interacts with users to translate natural language inputs into structured legal terminology.
- **NLP**: Natural Language Processing—AI-driven techniques for understanding and generating human language.
- **KVKK**: (Turkey’s Personal Data Protection Law).
- **GDPR**: General Data Protection Regulation—A comprehensive data protection law applicable in the European Union.
- **Simulation**: A virtual environment designed to replicate real-world scenarios for training purposes, such as mock trials or mediation sessions.
- **User Roles**: Specific categories of platform users, including clients, lawyers, and law students, each with tailored access and functionalities.
- **Two-Factor Authentication (2FA)**: A security process requiring two distinct forms of identification to verify user identity.

1.4 Overview

JustiWise represents a transformative approach to addressing inefficiencies in the legal sector and enhancing legal education. The platform integrates cutting-edge AI technologies with a well-designed and scalable architecture, delivering the following capabilities:

1. **Automated Legal Consultations**: Clients can describe their legal issues to an AI-powered avatar, which translates their input into structured legal terminology for review by lawyers.
2. **Interactive Training Modules**: Law students and professionals participate in realistic simulations of trials and mediations, gaining hands-on experience in a virtual environment.
3. **Secure and Compliant Operations**: JustiWise prioritizes data protection, ethical AI usage, and adherence to legal standards to ensure fairness and transparency.

By addressing persistent challenges and delivering innovative solutions, JustiWise aims to modernize the legal profession, expand access to justice, and enhance the quality of legal education for future generations.

2. Proposed software architecture

2.1 Overview

JustiWise's software architecture was created to include cutting-edge AI capabilities into an intuitive platform. To guarantee smooth communication between users and the system, the architecture uses a client-server model. The client interface, server, and database are the three primary parts of the system. Scalability, maintainability, and ease of development are made possible by this modular architecture.

Key features of the architecture include:

1.Client Interface :

A web-based front-end program, the client interface is made to be accessible on a variety of devices, such as smartphones, tablets, and PCs. To guarantee easy navigation and a flawless user experience, it places a strong emphasis on user-centric design concepts. Technologies like React.js are used in front-end development because of their adaptability and effectiveness in providing a consistent user experience across various devices. As previously mentioned in the project documentation, the interface uses responsive design and effective layout techniques to meet the needs of a wide range of users. Through dynamic forms, visual aids, and organized workflows, the client application makes it easier for users to interact with the system, increasing efficiency and usability.

2. AI-Powered Backend :

The backend, which uses Python to provide sophisticated AI capabilities and Java Spring Boot for effective server-side operations, is the foundation of the system's functioning. One essential element that makes it possible to convert user input into organized legal terminology is natural language processing, or NLP. This is accomplished by using a language model that is intended to interpret legal documents, examine user narratives, and produce pertinent answers. The backend also manages case-specific analysis, including the creation of mediation scenarios and document summaries. The modular architecture guarantees seamless connection with other system components and permits autonomous scaling of AI services. To preserve data integrity and user confidence, security processes are put in place, such as industry-standard methods for role-based authentication and encrypted API connection.

3. Database Layer :

The system aims to streamline legal workflows and provide a realistic training environment for law students and professionals.

JustiWise's data management system is built on the database layer. Because of its strong ability to handle structured data effectively and securely, a PostgreSQL database was selected. The database is made to hold a variety of data, such as training materials, legal papers, and user profiles. To guarantee data integrity and enhance query performance, advanced indexing and foreign key constraints are used. The database allows horizontal scaling to further improve scalability, enabling higher query throughput and storage capacity as user demand increases. Data flow between the components of the application is guaranteed by integration with the backend via secure RESTful APIs.

- Data Optimization: To save space and boost efficiency, redundant data storage is reduced.
- Security Features: Industry-standard techniques are used to encrypt sensitive data, including user-generated content and passwords. The system guarantees user privacy and data protection by adhering to legal criteria like KVKK and GDPR.
- Redundancy and Backup: Multi-region data replication and automatic daily backups protect against data loss, guaranteeing high availability and disaster recovery.
-
- Real-Time Updates: To give users the most recent information, including modifications to case statuses or new training modules, the database facilitates real-time synchronization.

2.2 Subsystem decomposition

The JustiWise platform consists of multiple subsystems, each dedicated to a specific function. These subsystems are designed to work collaboratively to deliver an efficient, scalable, and reliable solution.

Each subsystem communicates through RESTful APIs, allowing independent development and deployment while maintaining interoperability. Because of its modular design, the functionality of the platform as a whole is protected from being adversely affected by improvements or modifications made to one subsystem.

1.Subsystem for User Management:

- Gives clients, attorneys, and students role-based access control.
- Uses a safe method to handle user registration, authentication, and profile updates.
- Ensures that user data is managed and stored in accordance with data protection laws.

2.Subsystem for Legal Consultation:

- Includes an avatar driven by AI to communicate with clients and gather case data.
- Converts user-provided narratives into organized legal language so that attorneys can process them quickly.
- Keeps client-attorney interaction logs for future use and case monitoring.

3.Subsystem for Training and Simulation:

- Provides legal students with interactive modules to practice mediation and trial simulations.
- Offers tests and feedback in real time to help users become more proficient.
- Uses AI-generated dynamic scenarios and preloaded case studies to provide realistic training experiences.

4. Subsystem for Case Management:

- Permits attorneys to oversee case files, keep tabs on developments, and work together as a team.

- Integrates with the database to safely store and quickly retrieve legal papers.
- Provides automatic reminders, task scheduling, and document annotation to improve processes.

5. Subsystem for AI Integration:

- Uses sophisticated Natural Language Processing (NLP) to support automated consultations, evaluate legal papers, and compile case information.
- Makes it easier to integrate AI technologies for simulations of mediation and arbitration.
- Encourages ongoing education and model modifications in response to user input.

6. Subsystem for Support and Notification:

- Offers error-resolution and alerting tools to guarantee a continuous user experience.
- Includes a thorough helpdesk system for support requests and user questions.
- Notifies and updates users in real time on case statuses, training module outcomes, and account activity.

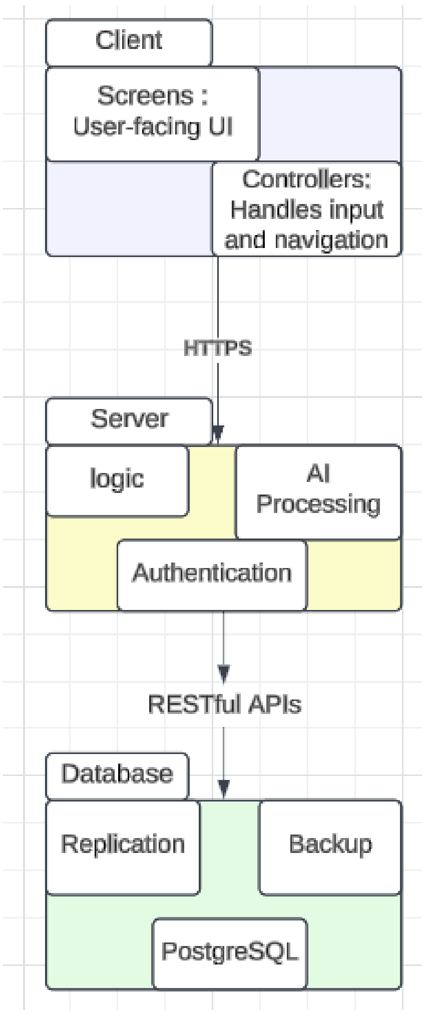
2.3 Hardware/software mapping

The hardware and software mapping of the JustiWise platform has been carefully planned to maximize security, scalability, and performance. The architecture makes use of a client-server approach in which the database and backend layers communicate with the front-end, which is located on client devices, via a secure network. Because the backend runs on a cloud-based infrastructure (like AWS or Google Cloud), it has high availability and can dynamically scale resources in response to changes in user demand.

The PostgreSQL-powered database layer is housed in the same cloud environment, facilitating quick and easy data retrieval. Data availability and resilience are guaranteed by sophisticated setups including automated backups and multi-region replication, which guarantee constant availability and safe access for all users.

Core features including AI-driven consultations, training simulations, and user authentication are integrated into the backend layer. The solution balances computational efficiency and extensibility by using technologies like Python for AI processing and Java Spring Boot for server-side functionality. RESTful APIs allow for communication between levels, and HTTPS protocols encrypt all data transfers. The infrastructure also facilitates GPU-accelerated instances for complex AI calculations, allowing training modules and legal narratives to be processed in real time.

This well-thought-out mapping approach guarantees that every part supports the system's main objectives of scalability, dependability, and user-centric operation.



2.4 Persistent data management

The foundation of JustiWise's operations is persistent data, which guarantees dependable and consistent performance across all capabilities. In order to effectively and safely manage data, the system takes an organized approach that is in line with the requirements of both law students and legal practitioners. The main tactics used for persistent data management are listed below:

Three types of data are mostly handled by the platform:

- **User Data**: Includes user profiles, roles, and identity-related information used for secure access management.
- **Legal Documents:** Contains client-uploaded case files as well as analysis and summary outputs produced by AI-powered technologies.
- **Educational Content:** Offers law students training simulations, user activity feedback, and more teaching resources.

JustiWise depends on a strong database architecture to provide smooth operations. The database, which is powered by PostgreSQL, effectively manages structured data and uses cutting-edge capabilities like:

- Indexing: Makes it easier to retrieve data quickly.

- Foreign Key Constraints: Preserves data integrity by facilitating the intricate connections seen in datasets related to law and education.

A key component of the platform's architecture is data security. Passwords and court case files, among other sensitive data, are protected by:

- Industry-standard algorithms are used in encryption to stop unwanted access.
- Regulatory Compliance: Respects KVKK and GDPR to protect user privacy and security.

The platform has extensive recovery and backup features, such as:

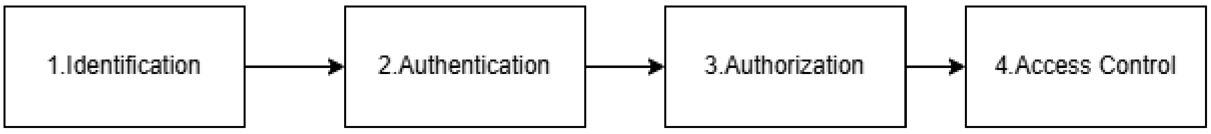
- Automated backups every day: Prevents data loss.
- In the event of a system breakdown, geographically distributed storage guarantees high availability and quick recovery.

Another essential component of JustiWise's long-term data management approach is scalability. The infrastructure of the database is:

- It can handle increasing data volumes as user demand rises since it is horizontally scalable.
- Optimized for Queries: Continues to function reliably even at high utilization levels.

JustiWise guarantees operational excellence and fosters user confidence by incorporating these strong data management techniques. The platform is a leading solution in the legal technology space because of its dedication to security, dependability, and scalability, which meets the changing needs of both students and legal professionals.

2.5 Access control and security



2.5.1 Identification

Initially, the system collects unique identifiers from the user, such as email addresses or usernames, as part of the registration process. The next step involves validating these inputs to ensure they are accurate and unique. Validation includes two checks:

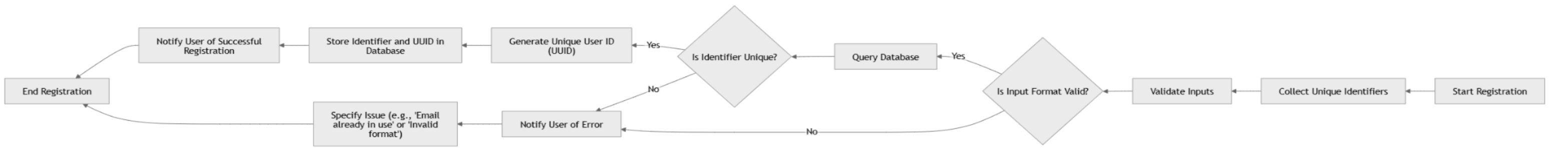
1. The system verifies the format of the input, such as ensuring the email matches a valid pattern.
2. The system queries the database to confirm the identifier is not already in use.

If the inputs pass validation:

- A unique identifier, known as a UUID (Universally Unique Identifier), is generated for the user to ensure their record can be uniquely identified in the system.
- The user's identifier, UUID, and associated roles or groups are securely stored in the database.
- The system sends a notification to the user, confirming successful registration.

If the validation fails:

- The system notifies the user of the issue, such as "Email already in use" or "Invalid format," and prompts them to correct the input.


Start Registration
Collect Unique Identifiers
Validate Inputs
Is Input Format Valid?
Yes
Query Database
Is Identifier Unique?
Yes
Generate Unique User ID (UUID)
Store Identifier and UUID in Database
Notify User of Successful Registration
End Registration
No
No
Notify User of Error
Specify Issue (e.g., 'Email already in use' or 'Invalid format')

2.5.2 Authentication

Password Authentication:

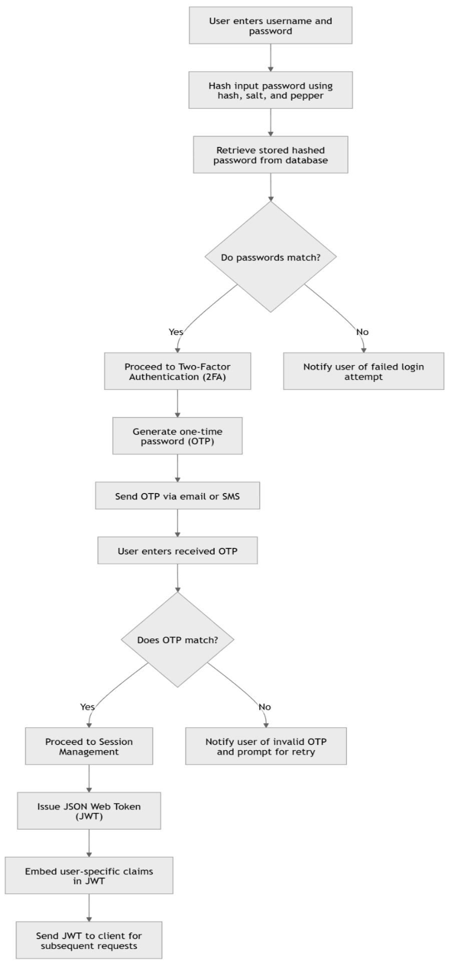
The user begins by entering their username and password. The system hashes the entered password using a combination of a hash function, salt (a unique value per user), and a secret pepper (stored securely). It retrieves the stored hashed password from the database and compares it to the hashed input password. If they match, the user is authenticated for this step; otherwise, the system rejects the login attempt.

Two-Factor Authentication (2FA):

For added security, the system generates a one-time password (OTP) and sends it to the user via email or SMS. The user then inputs the OTP into the system. The server validates the OTP by comparing it to the generated value. If the OTP matches, authentication proceeds; otherwise, the system prompts the user to retry or logs a failed attempt.

Session Management:

Once the user successfully passes 2FA, the system generates a JSON Web Token (JWT) to maintain the session. The JWT includes claims such as user roles, permissions, and an expiration time to ensure


User enters username and password
Hash input password using hash, salt, and pepper
Retrieve stored hashed password from database
Do passwords match?
Yes
No
Proceed to Two-Factor Authentication (2FA)
Notify user of failed login attempt
Generate one-time password (OTP)
Send OTP via email or SMS
User enters received OTP
Does OTP match?
Yes
No
Proceed to Session Management
Notify user of invalid OTP and prompt for retry
Issue JSON Web Token (JWT)
Embed user-specific claims in JWT
Send JWT to client for subsequent requests

secure session handling. The token is sent to the client, allowing the user to perform authenticated actions without re-entering credentials during the session.

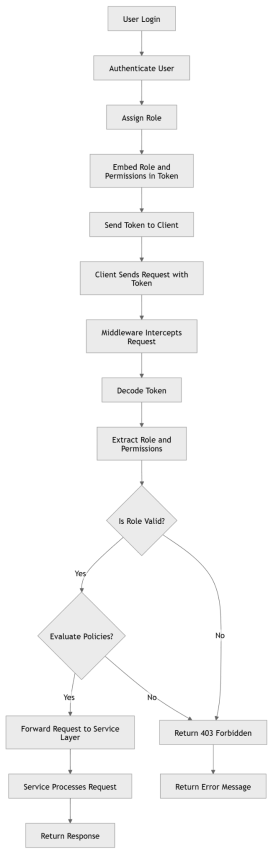
2.5.3 Authorization

Role-Based Access Control (RBAC):
In RBAC, users are assigned predefined roles such as client, lawyer,student and support. Each role has specific permissions defining what actions the user can perform or what resources they can access. Permissions are centrally managed, making it easier to enforce consistent access rules.

Policy-Based Access Control:
In addition to RBAC, policy-based access control evaluates more granular conditions (e.g., time of access, resource type) using a middleware layer. This middleware intercepts user requests, checks the user's role and additional conditions, and determines whether to allow or deny access.

Steps in the Process

1. **User Logs In**:
   - The user provides valid credentials.
   - Upon successful authentication, the server assigns the user a role and generates a token containing their role and permissions.
2. **Client Makes a Request**:
   - The client includes a jwt token in the request header.
3. **Middleware Intercepts the Request**:
   - Middleware extracts the token and decodes it.
   - Middleware checks the user's role and the associated permissions.
   - Middleware evaluates additional policies , such as resource type or access time.
4. **Decision Point**:
   - If the role and policies permit access, the request is forwarded to the appropriate service.
   - If not, the middleware denies the request and returns an error (e.g., 403 Forbidden).
5. **Response Handling**:
   - If access is granted, the service processes the request and returns the response.
   - If access is denied, the user receives an error message.


User Login
Authenticate User
Assign Role
Embed Role and Permissions in Token
Send Token to Client
Client Sends Request with Token
Middleware Intercepts Request
Decode Token
Extract Role and Permissions
Is Role Valid?
Yes
No
Evaluate Policies?
Yes
No
Forward Request to Service Layer
Return 403 Forbidden
Service Processes Request
Return Error Message
Return Response

2.5.4 Access Control

Role-Based Access Control (RBAC) is implemented to define and manage permissions based on user roles. Below is the RBAC design tailored for JustiWise:

Roles and Responsibilities:

The system defines distinct roles with specific access rights:

Client:

- Interacts with the AI-powered avatar to narrate cases.
- Accesses their case history and uploaded documents.
- Receives case updates from assigned lawyers.

Lawyer:

- Reviews client-provided cases and legal documents.
- Manages assigned cases, tracks progress, and schedules tasks.
- Uploads and downloads legal documents.
- Uses legal document analysis and generation tools.
- Participates in simulation training modules for skill enhancement.
- Provides feedback for system improvement.

Law Student:

- Accesses virtual training modules, including mock trials and mediation simulations.
- Receives feedback on performance during training.
- Views educational materials and resources.

Support Team:

- Responds to user queries and troubleshooting requests.
- Accesses error logs and diagnostic tools for resolving system issues.
- Limited access to user data for resolving specific cases.

Access Permissions

Feature/Resource	**Client**	**Lawyer**	**Law Student**	**Support Team**
User Registration	Create own	Create own	Create own	N/A
Login/Authentication	Access own	Access own	Access own	Limited
Case Narration	Full	N/A	N/A	N/A

Case Review & Management	N/A	Full	N/A	Limited
Document Upload/Download	Limited	Full	Limited	N/A
Legal Document Analysis	N/A	Full	Limited	N/A
Simulations and Training	N/A	Full	Full	Limited
Audit Logs	N/A	N/A	N/A	View limited
Error Notifications	View	View	View	Full
Support Requests	Submit	Submit	Submit	Full

2.6 Global software control

Global software control is a key aspect of the JustiWise platform, ensuring the coordinated operation of its subsystems, maintaining reliability, and delivering a seamless user experience. Below are the specific mechanisms and strategies used for global software control based on the design and requirements of the project.

1. Centralized Control and Coordination

- **Subsystem Orchestration**:
  A centralized control service manages the interactions between all subsystems, including user management, legal AI consultation, training modules, and case management. This ensures that tasks are performed in a coordinated and consistent manner..
- **Event-Driven Architecture**:
  The platform employs an event-driven model to handle asynchronous operations like sending notifications, triggering document updates, and logging user activities. Events are queued and processed to maintain system responsiveness and traceability.

2. Error Handling and Notifications

- **Real-Time Error Reporting**:
  The system uses pop-up notifications to inform users of errors (e.g., "Error 500: Internal Server Error"). Error messages include actionable information to guide users toward resolution.
- **Centralized Error Logging**:
  All errors and warnings are logged in a centralized system for analysis and debugging. These logs capture essential details such as timestamps, affected subsystems, and user actions.

- **Automatic Failover Mechanisms**:
  Critical services have redundancy built in. In the event of a subsystem failure, the system switches to backup instances to maintain continuity. For example, a backup server handles requests if the primary AI service encounters downtime.

3. Synchronization and State Management

- **Real-Time Data Synchronization**:
  Changes in one subsystem are immediately reflected in others to ensure consistency. For example, updates to case details are visible to both clients and assigned lawyers without delay.
- **Session Management**:
  The system uses JWT tokens to track user sessions, ensuring that session information is synchronized across subsystems. Tokens include claims such as user roles and permissions.

4. Monitoring and Alerts

- **System Monitoring**:
  Support team can view real-time metrics like active user sessions, error rates, and resource usage through a centralized dashboard.
- **Proactive Alerts**:
  Automated alerts notify the Support team of critical issues, such as multiple failed login attempts, high server load, or data access anomalies. Alerts are sent via email or SMS for immediate attention.

5. Security Enforcement

- **Access Control**:
  Role-Based Access Control (RBAC) ensures users can only access features and data relevant to their role (e.g., clients cannot access lawyer-specific tools).
- **Compliance**:
  The system enforces compliance with KVKK and GDPR by encrypting data, managing consent for data usage, and providing audit trails for data access.

6. Startup and Shutdown Procedures

- **Startup**:
  During system startup, essential services such as database connections, AI models, and authentication mechanisms are initialized in a specific sequence. System health checks ensure all subsystems are operational before accepting user requests.
- **Shutdown**:
  For planned maintenance, the system performs a graceful shutdown, saving all active sessions and notifying users in advance. Automated backup processes ensure no data loss during the shutdown.

7. Scalability and Load Balancing

- **Dynamic Scaling**:
  The system dynamically allocates resources based on user demand. For example, additional AI processing nodes are added during peak usage periods like virtual training sessions.
- **Load Balancing**:
  Requests are distributed across multiple servers to prevent bottlenecks and maintain high performance.

8. Redundancy and Recovery

- **Backup and Redundancy**:
  The system performs daily automated backups, with data replicated across multiple regions to prevent loss during failures.
- **Recovery Mechanisms**:
  In case of a failure, the system restores operations using the latest backups. Recovery processes prioritize critical services like authentication and case management.

2.7 Boundary conditions

Boundary conditions define how the JustiWise platform behaves during critical operational states such as startup, shutdown, failure recovery, and exceptional situations. These conditions ensure reliability, stability, and a seamless experience for users.

1. Startup Conditions

- **Service Initialization**:
  - All core subsystems, such as user management, legal AI services, training modules, and case management, are initialized in a defined sequence to ensure dependencies are correctly established.
  - Connections to external services, such as email/SMS gateways for Two-Factor Authentication (2FA), are verified.
- **Health Checks**:
  - Each subsystem undergoes self-diagnostic checks to confirm readiness. For instance, the database and AI models are tested for availability.
- **AI Model Loading**:
  - Pre-trained AI models, such as Llama 3 for natural language processing, are loaded and validated to ensure functionality.
- **System Activation**:
  - Once all services are operational, the system transitions to an active state, and users are notified of its availability.

2. Shutdown Conditions

- **Graceful Termination**:
  - Active user sessions and ongoing processes, such as document uploads or case updates, are saved to prevent data loss.
  - Users are notified in advance about planned maintenance or shutdown events via in-app messages or emails.
- **Subsystem Deactivation**:
  - Non-critical services are deactivated first, followed by critical components like the database and AI services.
- **Log Archiving**:
  - All system logs are archived to preserve a record of activities leading up to the shutdown.
- **Backup Operations**:
  - Automated backups are performed to ensure data integrity before the system is completely shut down.

3. Failure Recovery

- **Automatic Failover**:
  - Redundant systems take over when a primary subsystem fails, ensuring minimal disruption. For example, backup servers handle requests if the primary server is unavailable.
- **Data Recovery**:
  - The system restores data from the latest backups in case of corruption or loss.
- **User Notifications**:
  - Users are informed of service interruptions with clear timelines for recovery and actionable suggestions.

4. Exceptional Scenarios

- **High Traffic Spikes**:
  - Dynamic scaling mechanisms allocate additional resources, such as server instances, to handle increased loads.
  - Rate limiting prevents abuse while maintaining access for genuine users.
- **Unauthorized Access Attempts**:
  - Multiple failed login attempts trigger account lockouts and notify users. Suspicious activities are logged and flagged for administrative review.
- **External Service Failures**:
  - If a third-party service (e.g., email or SMS provider) fails, the system retries the operation with alternative providers or queues the task for later.

5. Edge Cases

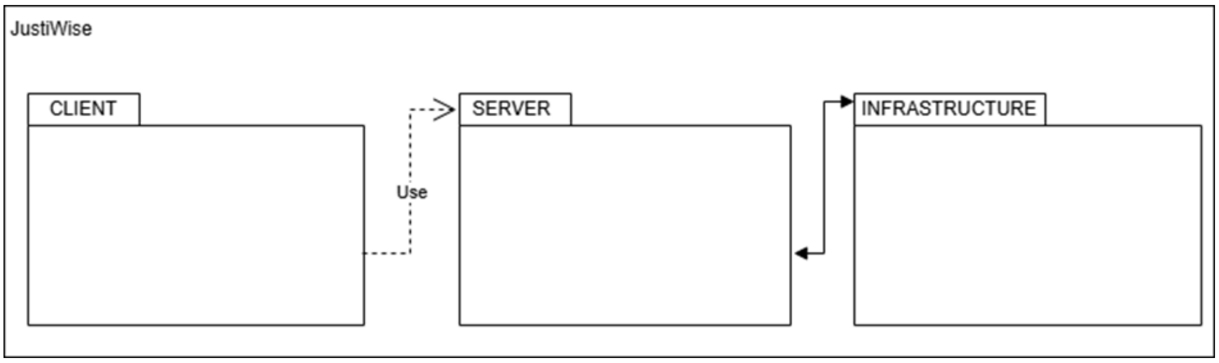
- **Interrupted Transactions**:
  - Incomplete processes, such as a partially uploaded document, are saved temporarily. Users can resume from where they left off after re-authentication.
- **Expired Sessions**:
  - Users are prompted to re-authenticate if their session token expires during an operation. Unsaved changes are preserved temporarily.
- **AI Model Errors**:
  - If the AI backend fails to process a user request, predefined templates or manual input options are provided as fallbacks.

6. User Notifications

- **Startup/Shutdown Notifications**:
  - Users receive messages about system availability or scheduled downtimes.
- **Failure Alerts**:
  - Real-time notifications inform users of issues such as failed uploads or service interruptions, along with estimated recovery times.

3. Subsystem services

The JustiWise platform is designed with a modular architecture to ensure scalability, maintainability, and clarity in its operations. The system is divided into three main subsystems, each dedicated to a specific functionality and working collaboratively to provide a flawless user experience. HTTP communication plays a vital role in how these subsystems interact. The diagram below illustrates the relationships between these subsystems:



a. Client Subsystem

- The Client subsystem represents the user-facing components of the platform. This includes all user interface elements, interaction modules, and controllers responsible for handling user inputs.
- Responsibilities:
    - Facilitating user interactions through screens such as login, profile management, and case dashboards.
    - Validating and processing user inputs via controllers like the Request Handler and Validation Controller.
    - Communicating with the Server Subsystem to send requests and retrieve results.

b. Server Subsystem

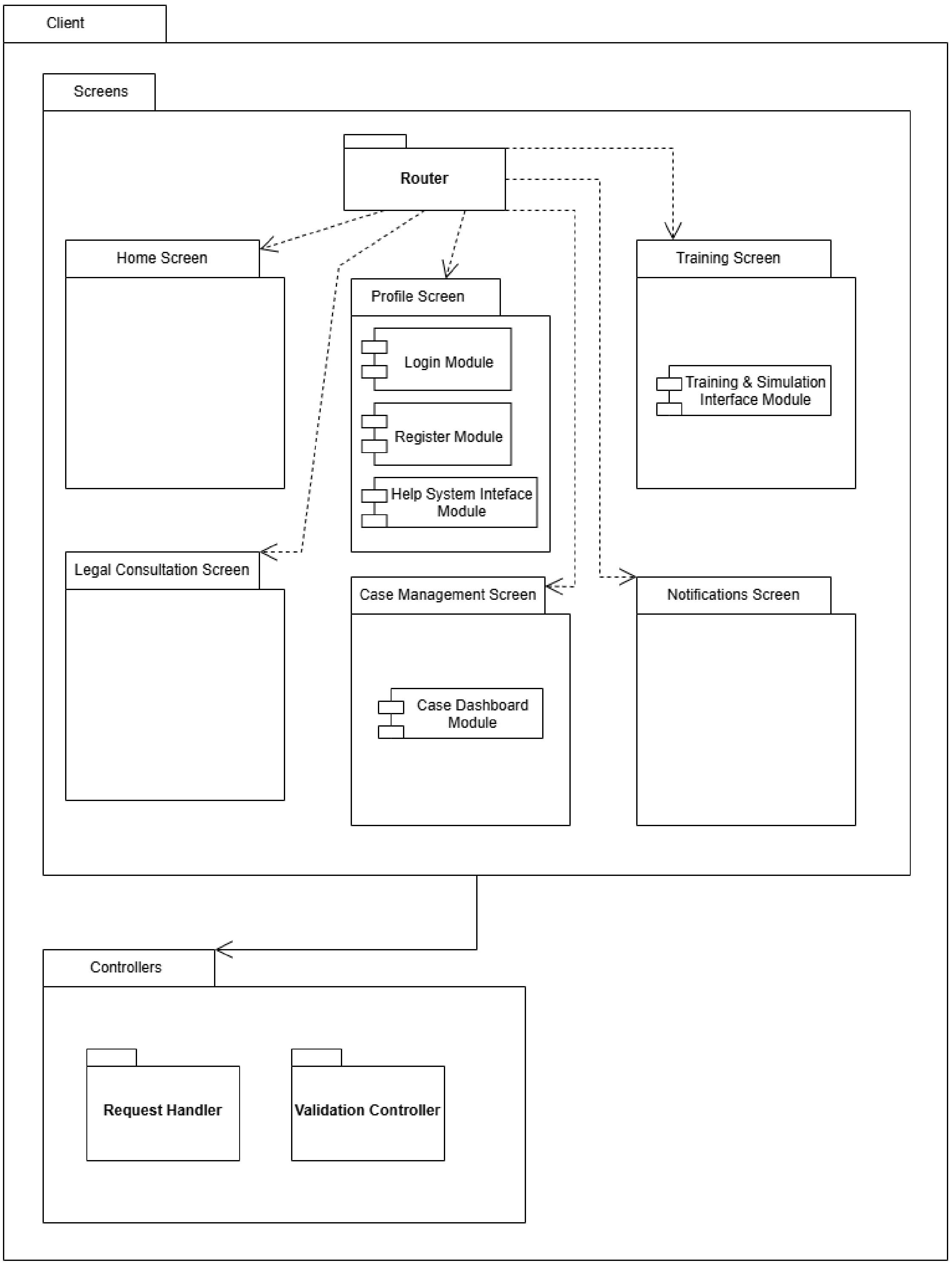
- The Server subsystem serves as the business logic layer, processing requests from the Client and interacting with the Infrastructure subsystem for data storage and retrieval.
- Responsibilities:
    - Handling authentication, case processing, training simulations, and AI-driven functionalities like NLP.
    - Coordinating with the Infrastructure subsystem to ensure data consistency and security.
    - Managing notifications and providing responses to the Client subsystem.

c. Infrastructure Subsystem

- The Infrastructure subsystem provides the backbone for data management, scalability, and performance optimization.
- Responsibilities:
    - Storing and managing data securely using a PostgreSQL database.
    - Handling backup and recovery operations to prevent data loss.
    - Ensuring system scalability and high performance, even under heavy loads.

The diagram represents the architecture of the Client-Side System for the JustiWise project, showing the relationship between the Router, Screens, and Controllers. Each component plays a specific role, ensuring modularity and maintainability. Below is a detailed explanation. The Client package serves as the main interface between the user and the system. It consists of the following sub-components:

3.1 Client



The hierarchy of modules inside the client system.

3.1.1 Screens

The **Screens** represent the user interface where users interact with the system. Each screen has a specific purpose and functionality:

- **Home Screen**:
    - Acts as the landing page, providing users with navigation options.
    - Connects with the Router for directing users to specific modules/screens.
- **Profile Screen**:
    - Allows users to manage their personal information and account settings.
    - Includes:
        - **Login Module**: Manages user authentication processes.
        - **Register Module**: Handles user registration and account creation.
        - **Help System Interface Module**: Offers guidance and support to users.
- **Training Screen**:
    - Provides access to training and simulation functionalities.
    - Includes:
        - **Training and Simulation Interface Module**: Manages interactive training modules.
- **Legal Consultation Screen**:
    - Facilitates AI-driven legal consultation for clients.
- **Case Management Screen**:
    - Allows lawyers to manage case files and related data.
    - Includes:
        - **Case Dashboard Module**: Displays and organizes case-related information.
- **Notifications Screen**:
    - Displays system notifications, alerts, and updates for the user.

The **Router** manages the navigation between screens. It serves as the central navigation mechanism, enabling users to move seamlessly between different screens based on their interactions.

- **Responsibilities**:
    - Directs users to the appropriate screen based on their actions.
    - Ensures smooth transitions between screens.

3.1.2 Controllers Package

The **Controllers** package serves as the intermediary between the screens and the server or backend modules. It ensures the proper handling of user inputs and system responses.

Request Handler

- **Purpose**:
    - Processes requests initiated by the user from the screens.
    - Forwards these requests to the appropriate server-side modules for further processing.
- **Examples**:
    - Submitting login credentials from the Login Screen.
    - Fetching case data for the Case Management Screen.

Validation Controller

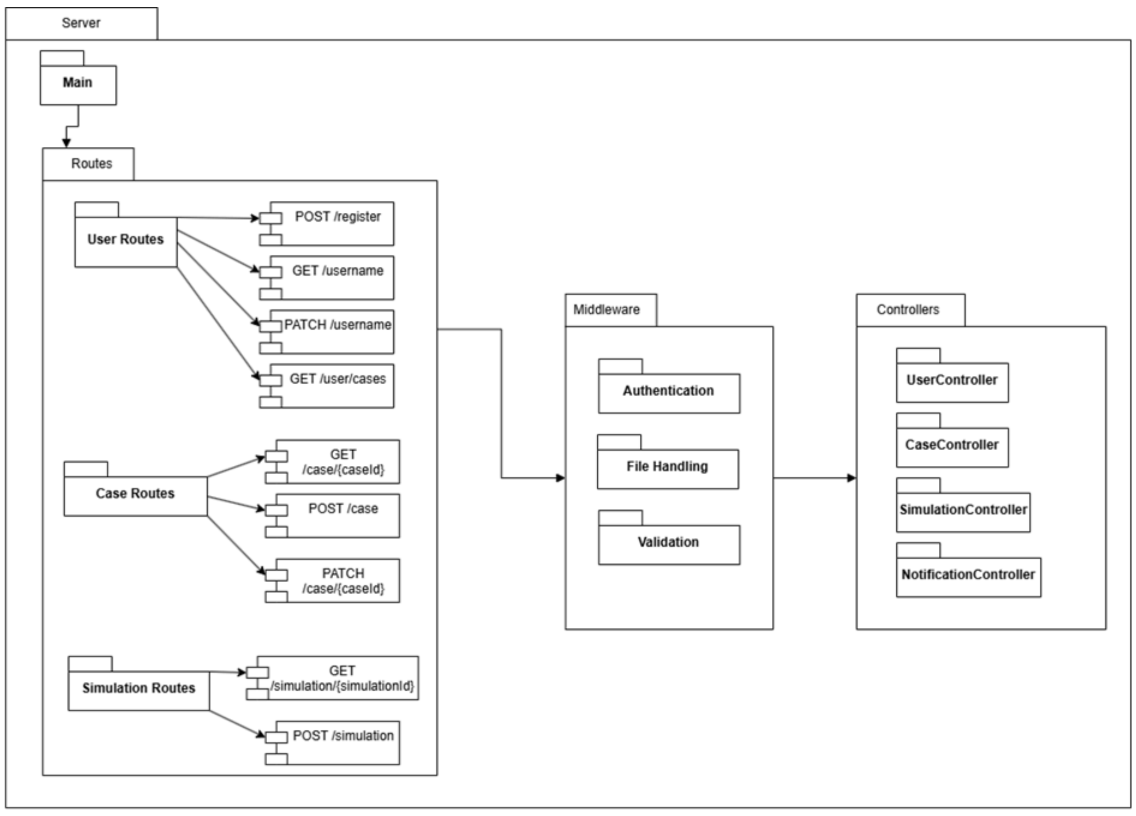
- **Purpose**:

- Validates user inputs to ensure data integrity and prevent errors.
    - Performs tasks such as checking for empty fields or validating email formats.
- **Examples**:
    - Validating form inputs on the Register Screen or Profile Screen.
    - Ensuring that all required fields are filled before sending data to the backend.

Relationships

- **Router → Screens**:
    - The **Router** connects directly to all screens, allowing users to navigate between them. This is represented with dependency arrows.
- **Screens → Controllers**:
    - Each screen connects to the Request Handler and/or Validation Controller based on its functionality.

3.2 Server



The hierarchy of layers and submodules within the server.

3.2.1 Main Package

The **Main** package is the starting point of the **Server** subsystem. It organizes and routes all requests from the **Client** side to the appropriate modules and controllers.

- **Functionality**:
  - The **Main** package routes incoming requests to the correct **Routes** package, which organizes them into user-related, case-related, and simulation-related routes.

3.2.2 Routes

Routes define the paths (URLs) that correspond to different actions or operations in the system. They serve as a bridge between the client's requests and the backend logic that processes those requests.

a. User Routes

These routes are dedicated to user-related actions. They include operations such as user registration, fetching user details, and updating user information.

- **POST /register**:
  - This route handles user registration by creating a new user in the system.
- **GET /username**:
  - This route fetches user details based on their username.
- **PATCH /username**:
  - This route updates the user's information based on the username.
- **GET /user/cases**:
  - Retrieves a list of cases associated with the user.

b. Case Routes

These routes handle actions related to legal cases, such as retrieving, creating, and updating cases.

- **GET /case/{caseId}**:
  - Retrieves specific case details by case ID.
- **POST /case**:
  - Creates a new case in the system.
- **PATCH /case/{caseId}**:
  - Updates details for an existing case.

c. Simulation Routes

These routes are responsible for managing legal simulations used for training or educational purposes.

- **GET /simulation/{simulationId}**:
  - Retrieves details about a specific simulation by its ID.
- **POST /simulation**:
  - Starts a new simulation session based on predefined data or user input.

3.2.3 Middleware

Middleware acts as a middle layer that processes requests before they reach the appropriate controllers. It performs essential tasks such as authentication, validation, and file handling.

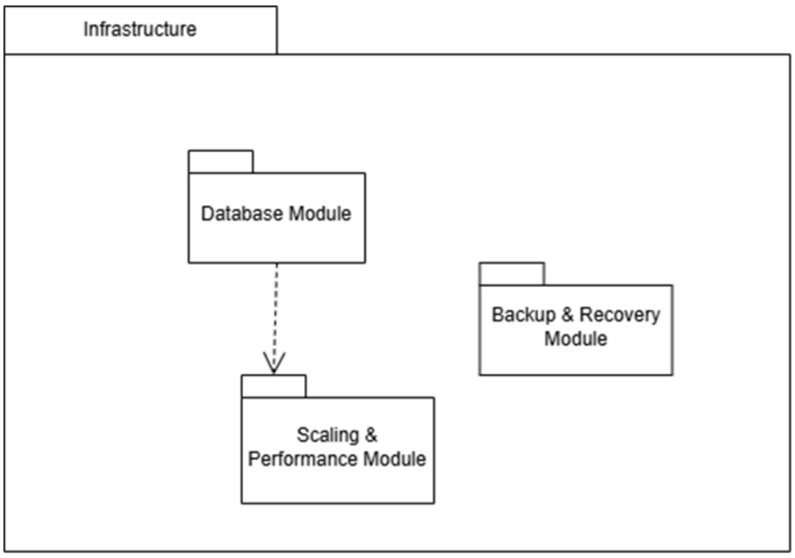
- **Authentication**:
  - Ensures that requests are made by authorized users. It validates tokens or session information to confirm the user's identity before proceeding to the controller.
- **File Handling**:
  - Manages file uploads and downloads, ensuring proper handling and storage of files (legal documents, case files).
- **Validation**:
  - Ensures that data coming from the client (such as form inputs or requests) is valid before reaching the controller. It checks for missing fields, incorrect data formats.

3.2.4 Controllers

Controllers contain the logic to process the requests sent from the routes and interact with the data layer. They contain the actual business logic and are responsible for sending the appropriate response back to the client.

- **UserController**:
  - Handles user-related operations, such as registering a new user, fetching user details, and updating user information.
  - **Functions**:
    - Add User, Get User, Update User.
- **CaseController**:
  - Manages case-related functionalities, including creating a new case, retrieving case details, and updating case status.
  - **Functions**:
    - Get Case, Add Case, Update Case.
- **SimulationController**:
  - Handles operations related to legal simulations, such as fetching simulation details or starting a new simulation session.
  - **Functions**:
    - Get Simulation, Start Simulation.
- **NotificationController**:
  - Responsible for sending notifications to users about important events like case updates or simulation results.
  - **Functions**:
    - Send Notification, Get Notifications.

3.3 Infrastructure



1. Database Module

- **Purpose**: Handles all data interactions, storing and managing data for users, cases, notifications, and simulations.
- **Key Features**:
    - **Data Management**: Stores user profiles, case details, and notifications.
    - **Data Security**: Ensures encrypted and secure data.
    - **Performance Optimization**: Manages database indexing for faster retrieval.
- **Relationships**:
    - **Scaling & Performance Module → Database Module**: Ensures optimized data storage and retrieval.

2. Backup & Recovery Module

- **Purpose**: Ensures data availability and safety through automated backups and recovery processes.
- **Key Features**:
    - **Regular Backups**: Periodic backups to prevent data loss.
    - **Data Recovery**: Recovery mechanisms in case of failure.
    - **Disaster Recovery**: System recovery after major failures.
- **Relationships**:
    - **Backup & Recovery Module → Database Module**: Collaborates with the database for backup and restoration.

3. Scaling & Performance Module

- **Purpose**: Ensures the system can handle growing data and traffic while maintaining performance.
- **Key Features**:
    - **Load Balancing**: Distributes traffic to avoid server overload.
    - **Database Scaling**: Optimizes data handling via sharding or partitioning.

    - **Caching**: Improves response times using caching technologies like Redis.
    - **Performance Monitoring**: Continuously monitors and optimizes system performance.
- **Relationships**:
    - **Scaling & Performance Module → Database Module**: Optimizes database scalability.
    - **Scaling & Performance Module → Backup & Recovery Module**: Ensures efficient backup and recovery performance.

4. Glossary

AI (Artificial Intelligence): Technology designed to simulate human intelligence for complex tasks such as decision-making, language processing, and pattern recognition.

Avatar: A digital entity that interacts with users to translate natural language inputs into structured legal terminology.

Case Management: A systematic process within JustiWise for organizing, tracking, and updating legal case files to improve efficiency and accessibility for lawyers.

Client: An individual user of the JustiWise platform seeking legal advice or services.

Compliance: Adherence to legal standards such as KVKK (Turkey's Personal Data Protection Law) and GDPR (General Data Protection Regulation), ensuring data privacy and ethical AI usage.

GDPR (General Data Protection Regulation): A comprehensive data protection regulation implemented in the European Union to safeguard personal data and privacy.

KVKK: Turkey's Personal Data Protection Law that regulates how personal data is collected, processed, and stored to protect user privacy.

Legal Consultation: The process of providing users with AI-driven legal advice, including document preparation, case analysis, and personalized recommendations.

Legal Simulation: Virtual training modules within JustiWise that allow law students and professionals to participate in mock trials, mediation, and arbitration scenarios.

Natural Language Processing (NLP): AI-driven techniques enabling systems to understand, interpret, and generate human language for tasks like legal document analysis.

PostgreSQL: A powerful, open-source relational database system used in JustiWise for secure and efficient data storage and management.

RBAC (Role-Based Access Control): A security mechanism that restricts system access based on user roles, ensuring that users interact with features relevant to their responsibilities.

Simulation: An interactive virtual environment replicating real-world legal scenarios to provide practical training experiences for law students and professionals.

Two-Factor Authentication (2FA): A security process requiring two forms of verification—such as a password and a one-time code—to ensure secure user authentication.

User Roles: Categories of JustiWise platform users, including clients, lawyers, law students, and support staff, with tailored access to system features.

5. References

[1] PERSONAL DATA PROTECTION INSTITUTION | KVKK | Personal Data Protection Authority. (2024). Kvkk.gov.tr. https://kvkk.gov.tr/

[2] The Code affirms an obligation of computing professionals to use their skills for the benefit of society. (n.d.). Www.acm.org. https://www.acm.org/code-of-ethics/

[3] IEEE - IEEE Code of Ethics. (n.d.). https://www.ieee.org/about/corporate/governance/p7-8.html

[4]"Design Pattern Quick Guide - Tutorialspoint," *www.tutorialspoint.com*. https://www.tutorialspoint.com/design_pattern/design_pattern_quick_guide.htm