Intelligent Academic Problem-Solving System

(IAPSS)

Software Architecture & Security Documentation

Author

Hemant Verma

Live Application

`https://iapss.vercel.app/`

Source Code Repository

`https://github.com/Hemant-Verma-IND/iapssfinal`

Version: v1.0
Last Updated: January 16, 2026

Preface

Modern academic and technical learning environments increasingly rely on intelligent digital systems to assist users in understanding, analysing, and solving complex problems. However, most existing solutions prioritise output generation over structured reasoning, often failing to support users in developing a clear understanding of the underlying concepts and logic.

The **Intelligent Academic Problem-Solving System (IAPSS)** is designed to address this gap. The system focuses on analytical assistance rather than direct answer delivery, enabling users to deconstruct academic problem statements and source code in a systematic and meaningful way. The core philosophy of the project is to promote clarity, explainability, and responsible use of intelligent systems in educational contexts.

This document serves as the official software documentation for IAPSS. It captures the system's background, architectural design, functional and non-functional requirements, security considerations, and deployment model. The documentation is intended for future reference, maintenance, onboarding of contributors, and long-term evolution of the platform.

Certain features discussed in this document are intentionally scoped as future enhancements. These are clearly separated from the current implementation to maintain transparency between the system's present capabilities and its planned evolution.

The documentation emphasises strong architectural foundations and security-aware design decisions, ensuring that the system remains extensible, maintainable, and ethically aligned as it evolves over time.

Document Control

Document Title	Intelligent Academic Problem-Solving System – Software Documentation
Author	Hemant Verma
Project Code Name	IAPSS
Document Version	v1.0
Document Status	Active
Last Updated	January 16, 2026
Intended Audience	Developers, Maintainers, Future Contributors
Confidentiality	Public (Academic / Reference Use)

Contents

Chapter 1

Introduction

Problem-solving is a core skill in technical education, particularly in domains such as mathematics, data structures, algorithms, and competitive programming. While a vast amount of learning material and problem repositories are available, learners often struggle to identify meaningful guidance that helps them improve their reasoning process rather than merely obtaining correct solutions.

During preparation for competitive programming and similar problem-intensive activities, learners frequently seek structured tips, optimisation strategies, and feedback on their approach. Existing tools often provide either complete solutions or isolated hints, which may not effectively support progressive skill development or self-assessment. This gap becomes more evident when users wish to understand *why* a solution works, how it can be improved, or what alternative approaches could be considered.

The **Intelligent Academic Problem-Solving System (IAPSS)** is designed to address these challenges by offering analytical assistance focused on problem interpretation, logical breakdown, and improvement-oriented insights. Instead of functioning as a traditional solution generator, the system aims to support users in refining their problem-solving strategies, identifying common pitfalls, and improving code quality and efficiency.

The platform is built with a strong emphasis on explainability, modularity, and responsible use of intelligent systems. It supports both academic problem analysis and source code evaluation, making it suitable for learners preparing for competitive programming, coursework, and self-directed technical practice.

This chapter establishes the foundational motivation and design philosophy of IAPSS, setting the context for the architectural, functional, and security-related decisions discussed in subsequent sections.

Chapter 2

System Overview

The Intelligent Academic Problem-Solving System (IAPSS) is a web-based, full-stack application designed to assist users in analysing academic problems and source code in a structured and explainable manner. The system is intended to support learning, self-improvement, and analytical reasoning rather than direct solution generation.

IAPSS operates as an interactive platform where users can submit textual problem statements, source code, or related inputs and receive structured analytical feedback. The system focuses on understanding problem intent, breaking down logic, highlighting improvement areas, and offering guidance-oriented insights relevant to academic learning and competitive programming preparation.

2.1 System Purpose

The primary purpose of IAPSS is to:

- Assist users in understanding complex problem statements.
- Provide analytical breakdowns rather than final answers.
- Offer improvement-oriented guidance for problem-solving and coding practices.
- Encourage ethical and responsible use of intelligent systems in learning.

2.2 System Scope

The current scope of the system includes:

- Analysis of academic problem statements.
- Source code analysis with focus on logic, structure, and optimisation hints.
- User authentication and access-controlled features.
- Web-based interaction through a modern frontend interface.

The system does not aim to:

- Act as an automatic solution generator.
- Replace formal learning or instructional material.
- Guarantee correctness or optimality of all suggestions.

2.3 User Roles

The system currently supports the following user role:

- **Authenticated User**: A registered user who can access protected features such as problem analysis, code analysis, and history-related functionalities.

Future versions may introduce additional roles such as administrators or moderators; these are documented under future scope.

2.4 Operating Environment

IAPSS is designed to operate in a standard web environment with the following assumptions:

- Users access the system via modern web browsers.
- The frontend and backend communicate through secure HTTP-based APIs.
- The system relies on external services for intelligent analysis where applicable.

This overview establishes a clear boundary for the system's responsibilities and provides a foundation for the architectural and security decisions discussed in subsequent chapters.

Chapter 3

Overall Architecture

The Intelligent Academic Problem-Solving System (IAPSS) follows a modular, layered architecture that separates user interaction, application logic, and external services. This design ensures scalability, maintainability, and clear security boundaries between system components.

The architecture is intentionally designed to be service-oriented and stateless, enabling future extensions such as additional analysis modules, role-based access control, and alternative frontend clients.

3.1 Architectural Style

IAPSS adopts the following architectural principles:

- **Client–Server Architecture**: Clear separation between frontend and backend responsibilities.
- **Stateless API Design**: Each request contains all required context, improving scalability.
- **Modular Components**: Independent modules for authentication, analysis, and data handling.
- **Security-by-Boundary**: Authentication and protected routes enforce access control.

3.2 High-Level Architecture

At a high level, the system consists of:

- A web-based frontend responsible for user interaction.
- A backend API layer handling authentication, request validation, and orchestration.
- Intelligent analysis services accessed via controlled interfaces.
- Persistent storage for user-related metadata where applicable.

3.3 Architecture Diagram

The following diagram represents the high-level architecture of IAPSS using Mermaid notation.

```
flowchart TD
    User[User Browser]
    Frontend[Frontend Web Application]
    Backend[Backend API Server]
    Auth[Authentication Module]
    Analysis[Analysis Services]
    DB[(Data Store)]

    User --> Frontend
    Frontend -->|API Requests| Backend
    Backend --> Auth
    Backend --> Analysis
    Backend --> DB
    Auth --> Backend
    Analysis --> Backend
```

3.4 Rendered Architecture Diagram

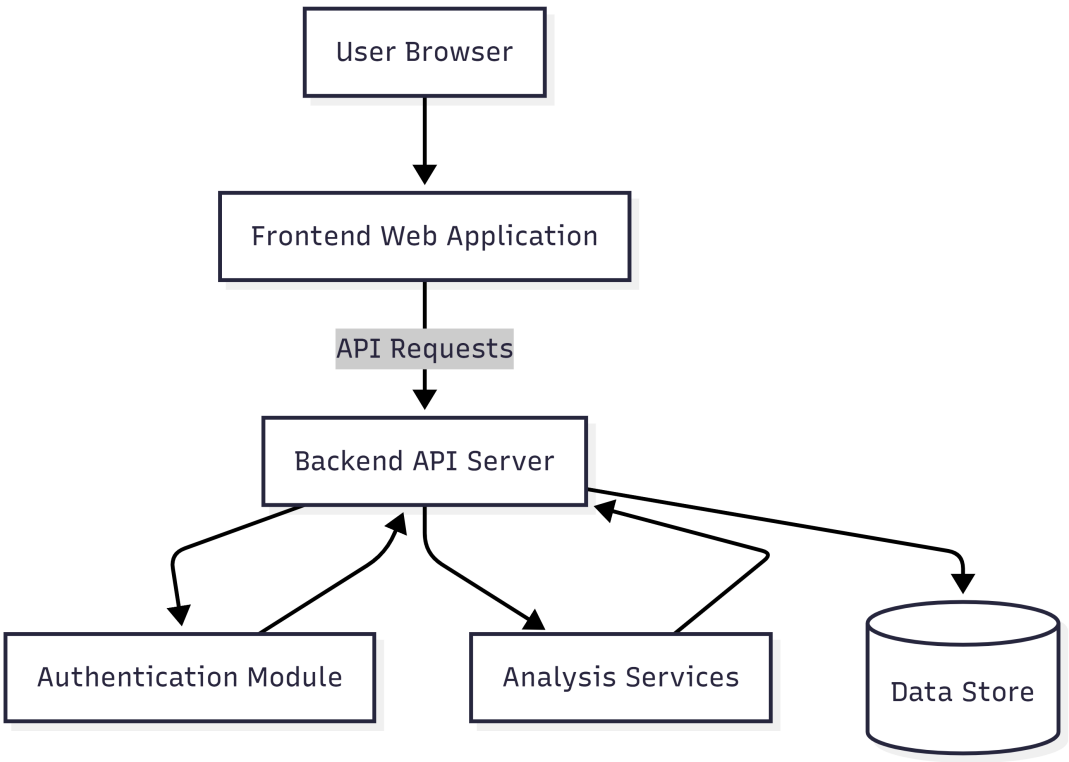

Figure 3.1: High-Level System Architecture of IAPSS

3.5 Component Responsibilities

3.5.1 Frontend Application

The frontend is responsible for:

- User interface and interaction.
- Input collection for problems and source code.
- Route-level access control for protected views.
- Communicating with backend APIs over HTTP.

No sensitive logic or credentials are stored on the client side.

3.5.2 Backend API Server

The backend serves as the central control layer and is responsible for:

- Request validation and sanitisation.
- Authentication and token verification.
- Routing requests to appropriate analysis services.
- Enforcing access control policies.

3.5.3 Authentication Module

The authentication module manages:

- User identity verification.
- Token issuance and validation.
- Protection of restricted endpoints.

3.5.4 Analysis Services

Analysis services provide:

- Problem interpretation and breakdown.
- Code structure and improvement analysis.
- Guidance-oriented feedback rather than direct solutions.

These services are accessed through controlled interfaces to prevent misuse.

3.5.5 Data Storage

Persistent storage is used selectively to store:

- User metadata.
- Request history where applicable.

Sensitive or confidential academic data is intentionally minimised.

3.6 Security Boundaries

Security boundaries are enforced at:

- Frontend route protection level.
- Backend API authentication middleware.
- Controlled access to analysis services.

This layered approach ensures that unauthorised access attempts are intercepted early in the request lifecycle.

Chapter 4

Functional Requirements

This chapter defines the functional requirements of the Intelligent Academic Problem-Solving System (IAPSS). The requirements specified in this section describe the observable behaviour of the system and the functionalities currently implemented in the active version.

Each requirement is uniquely identified to support traceability, testing, and future enhancement planning.

4.1 User Authentication

FR-01: User Registration

- The system shall allow users to create an account using valid credentials.
- The system shall validate input data before account creation.
- The system shall prevent duplicate registrations using the same identifier.

FR-02: User Login

- The system shall allow registered users to authenticate using valid credentials.
- The system shall issue an authentication token upon successful login.
- The system shall reject invalid authentication attempts.

FR-03: Session Management

- The system shall maintain authenticated user sessions using token-based authentication.
- The system shall restrict access to protected resources for unauthenticated users.

4.2 Access Control

FR-04: Protected Routes

- The system shall restrict access to analysis features to authenticated users.
- The system shall redirect unauthenticated users to the login interface.

4.3 Problem Analysis

FR-05: Problem Statement Submission

- The system shall allow users to submit textual academic problem statements.
- The system shall validate submitted input for format and length.

FR-06: Problem Analysis Processing

- The system shall analyse submitted problem statements.
- The system shall generate structured analytical feedback.
- The system shall avoid providing direct final answers.

4.4 Code Analysis

FR-07: Source Code Submission

- The system shall allow users to submit source code for analysis.
- The system shall accept a supported programming language identifier.

FR-08: Code Analysis Processing

- The system shall analyse submitted source code for structure and logic.
- The system shall provide improvement-oriented feedback.
- The system shall highlight potential optimisation opportunities where applicable.

4.5 History Management

FR-09: Analysis History Storage

- The system shall store metadata related to user analysis requests.
- The system shall associate stored history with the authenticated user.

FR-10: History Retrieval

- The system shall allow users to view previous analysis records.
- The system shall prevent unauthorised access to another user's history.

4.6 Error Handling

FR-11: Input Validation Errors

- The system shall detect invalid or malformed inputs.
- The system shall return meaningful error messages to the user.

FR-12: Service Failure Handling

- The system shall handle backend or analysis service failures gracefully.
- The system shall notify users when a request cannot be processed.

4.7 System Feedback

FR-13: User Feedback Presentation

- The system shall present analysis results in a readable and structured format.
- The system shall ensure clarity and consistency in feedback presentation.

This chapter defines the functional baseline of the system. Features not explicitly listed here are considered out of scope for the current implementation and are addressed separately under future enhancements.

Chapter 5

Non-Functional Requirements

This chapter defines the non-functional requirements of the Intelligent Academic Problem-Solving System (IAPSS). These requirements describe the quality attributes, constraints, and operational characteristics of the system rather than specific behaviours.

5.1 Performance Requirements

NFR-01: Response Time

- The system shall provide analysis responses within an acceptable time frame under normal usage conditions.
- User interface interactions shall remain responsive during request processing.

NFR-02: Concurrent Usage

- The system shall support multiple users accessing the platform concurrently.
- Performance degradation under moderate concurrent load shall be minimised.

5.2 Security Requirements

NFR-03: Authentication Security

- The system shall protect authenticated endpoints using token-based authentication.
- Authentication tokens shall not be exposed through client-side logic.

NFR-04: Access Control

- The system shall restrict access to protected features based on authentication state.
- Unauthorised access attempts shall be rejected at the API boundary.

NFR-05: Data Protection

- The system shall minimise storage of sensitive user data.
- User-submitted content shall not be publicly accessible by default.

NFR-06: Secure Communication

- All client–server communication shall occur over secure HTTP channels.
- Direct access to internal services shall be prevented.

5.3 Reliability Requirements

NFR-07: Fault Tolerance

- The system shall handle service errors gracefully without crashing.
- Partial failures shall not compromise overall system stability.

NFR-08: Availability

- The system shall be available during normal operational periods.
- Planned downtime shall be minimised and controlled.

5.4 Scalability Requirements

NFR-09: Horizontal Scalability

- The backend architecture shall support horizontal scaling.
- Stateless API design shall enable load distribution.

5.5 Maintainability Requirements

NFR-10: Modular Design

- The system shall follow modular design principles.
- Changes to one module shall not require extensive modifications to others.

NFR-11: Code Readability

- The codebase shall follow consistent coding standards.
- Documentation shall support future maintenance and onboarding.

5.6 Usability Requirements

NFR-12: User Interface Clarity

- The system shall present a clear and intuitive user interface.
- Feedback and error messages shall be understandable to users.

5.7 Ethical and Usage Constraints

NFR-13: Responsible Use

- The system shall encourage responsible academic usage.
- The system shall not promote unethical academic practices.

This chapter establishes the quality and security expectations of the system and provides a foundation for evaluating system behaviour beyond functional correctness.

Chapter 6

Module Design

This chapter describes the internal module-level design of the Intelligent Academic Problem-Solving System (IAPSS). Each module is defined by its responsibilities, interactions, and boundaries within the system architecture.

The modular design approach ensures separation of concerns, improved maintainability, and ease of future extension.

6.1 Authentication Module

6.1.1 Purpose

The Authentication Module is responsible for verifying user identity and controlling access to protected system resources.

6.1.2 Responsibilities

- User login and authentication.
- Issuance and validation of authentication tokens.
- Enforcement of access restrictions on protected routes.

6.1.3 Interactions

- Communicates with the frontend to validate user credentials.
- Interacts with the backend API to verify authentication state.

6.2 Frontend Access Control Module

6.2.1 Purpose

This module ensures that protected views are accessible only to authenticated users.

6.2.2 Responsibilities

- Verification of authentication state on route access.
- Redirection of unauthenticated users to login interfaces.

6.2.3 Interactions

- Reads authentication state provided by the backend.
- Acts as a security boundary at the user interface layer.

6.3 Problem Analysis Module

6.3.1 Purpose

The Problem Analysis Module processes academic problem statements and provides structured analytical feedback.

6.3.2 Responsibilities

- Acceptance of problem statement inputs.
- Validation and preprocessing of submitted data.
- Invocation of analysis services.
- Formatting of analytical feedback.

6.3.3 Interactions

- Receives inputs from the frontend interface.
- Communicates with backend analysis services.

6.4 Code Analysis Module

6.4.1 Purpose

The Code Analysis Module evaluates submitted source code and provides improvement-oriented insights.

6.4.2 Responsibilities

- Acceptance of source code and language metadata.
- Validation of code input.
- Analysis of code structure and logic.
- Generation of feedback related to readability and optimisation.

6.4.3 Interactions

- Interfaces with the backend API for analysis processing.
- Returns structured feedback to the frontend.

6.5 History Management Module

6.5.1 Purpose

The History Management Module maintains records of user interactions for reference and continuity.

6.5.2 Responsibilities

- Storage of analysis metadata linked to users.
- Retrieval of historical analysis records.

6.5.3 Interactions

- Communicates with persistent storage through the backend.
- Enforces access control on stored data.

6.6 Error Handling Module

6.6.1 Purpose

The Error Handling Module ensures graceful handling of invalid inputs and system failures.

6.6.2 Responsibilities

- Detection of invalid or malformed requests.
- Generation of meaningful error responses.
- Prevention of system crashes due to unexpected conditions.

6.6.3 Interactions

- Operates across all backend modules.
- Provides feedback to frontend components.

This modular design ensures clear responsibility boundaries and supports incremental enhancement of the system without large-scale refactoring.

Chapter 7

API Contracts

This chapter describes the backend API interfaces used by the Intelligent Academic Problem-Solving System (IAPSS). The documented endpoints represent stable and actively used services. APIs planned for future development are excluded and documented separately under future scope.

7.1 General API Conventions

- All APIs follow a REST-style design.
- Requests and responses use JSON format unless specified otherwise.
- Protected endpoints require a valid authentication token.
- Error responses follow a consistent structure.

7.2 Authentication APIs

7.2.1 User Login

Endpoint: `POST /api/auth/login`

- **Description:** Authenticates a user and issues an authentication token.
- **Authentication Required:** No

Request Body:

```
{
  "email": "user@example.com",
  "password": "********"
}
```

Response:

```
{
  "token": "jwt_token",
  "user": {
    "id": "user_id",
    "email": "user@example.com"
  }
}
```

7.2.2 User Registration

Endpoint: `POST /api/auth/register`

- **Description:** Registers a new user.
- **Authentication Required:** No

7.3 Problem Analysis APIs

7.3.1 Analyse Problem Statement

Endpoint: `POST /problems/analyse`

- **Description:** Analyses a submitted academic problem statement.
- **Authentication Required:** Yes

Request Body:

```
{
  "text": "Problem statement text"
}
```

Response:

```
{
  "analysis": "Structured analytical feedback"
}
```

7.4 Code Analysis APIs

7.4.1 Analyse Source Code

Endpoint: `POST /code/analyse`

- **Description:** Analyses submitted source code.
- **Authentication Required:** Yes

Request Body:

```
{
  "code": "source code",
  "language": "programming_language"
}
```

Response:

```
{
  "analysis": "Code analysis feedback"
}
```

7.5 History APIs

7.5.1 Save Code Analysis History

Endpoint: `POST /api/code/save`

- **Description:** Saves metadata related to code analysis.
- **Authentication Required:** Yes

7.6 Error Response Format

All error responses follow the structure:

```
{
  "error": "Error description message"
}
```

This standardised format ensures consistent error handling across the frontend and backend.

Chapter 8

Security Considerations

Security is treated as a core design concern in the Intelligent Academic Problem-Solving System (IAPSS) rather than an afterthought. The system is designed with the assumption that any publicly accessible interface may be subject to misuse, unintended access, or malicious probing. As a result, security boundaries are enforced consistently across architectural layers.

This chapter documents the security principles, controls, and limitations of the current system implementation.

8.1 Security Design Philosophy

The security design of IAPSS is guided by the following principles:

- **Least Privilege**: Access to system resources is restricted to authenticated users only.
- **Trust Boundaries**: Client-side components are treated as untrusted by default.
- **Fail-Safe Defaults**: Requests are denied unless explicitly permitted.
- **Transparency of Limitations**: Security controls are clearly scoped and documented.

The system intentionally avoids security-through-obscurity and instead relies on explicit validation and access control.

8.2 Authentication and Authorisation

8.2.1 Token-Based Authentication

IAPSS uses token-based authentication to manage user sessions:

- Authentication tokens are issued only after successful credential verification.
- Tokens are required for accessing protected API endpoints.
- Backend services validate tokens on every protected request.

Sensitive authentication logic is handled exclusively on the server side.

8.2.2 Route-Level Protection

- Protected frontend routes enforce authentication checks before rendering.
- Backend endpoints independently verify authentication status, ensuring defence-in-depth.

This dual-layer protection prevents reliance on client-side enforcement alone.

8.3 Input Validation and Sanitisation

- All user inputs are validated at the API boundary.
- Unexpected or malformed requests are rejected early.
- The system avoids executing or directly trusting user-submitted content.

This reduces the risk of injection attacks, malformed payload abuse, and unintended execution paths.

8.4 API Security

- APIs follow a stateless design, reducing session fixation risks.
- Protected endpoints require explicit authentication headers.
- Error responses avoid leaking internal implementation details.

Internal services are not directly exposed to the public network.

8.5 Data Handling and Privacy

- The system minimises storage of sensitive user data.
- User-submitted academic content is treated as transient wherever possible.
- Access to stored metadata is strictly scoped to the authenticated user.

The system is not designed to store confidential or personally sensitive information.

8.6 Threat Awareness and Limitations

While reasonable security practices are implemented, the system acknowledges the following limitations:

- The platform is not intended for high-risk or production-critical deployments.
- Advanced attack vectors such as coordinated denial-of-service attacks are outside the current scope.
- Security controls are designed for academic and controlled usage environments.

These limitations are documented to ensure realistic expectations and responsible use.

8.7 Responsible Use Policy

IAPSS is designed to promote ethical academic assistance. The system:

- Avoids direct solution disclosure.
- Encourages analytical understanding over answer replication.
- Discourages misuse for unethical academic practices.

This alignment between security and ethical usage ensures that the system remains both technically and academically responsible.

This chapter establishes security as a foundational system concern and provides a transparent view of the protections and constraints present in the current implementation.

Chapter 9

Deployment Architecture

This chapter describes the deployment model of the Intelligent Academic Problem-Solving System (IAPSS). The deployment architecture is designed to ensure separation of concerns, scalability, and controlled exposure of system components.

The system is deployed as a distributed web application with independently hosted frontend and backend services.

9.1 Deployment Overview

IAPSS follows a two-tier deployment model:

- A publicly accessible frontend web application.
- A backend API service exposed only through defined endpoints.

This separation ensures that user-facing components and core application logic are isolated, reducing attack surface and improving maintainability.

9.2 Frontend Deployment

The frontend application is deployed as a static web application using a cloud-based hosting platform.

- **Deployment URL:** `https://iapss.vercel.app/`
- **Hosting Type:** Static frontend hosting with CDN support

9.2.1 Frontend Responsibilities in Deployment

- Rendering user interfaces and handling user interaction.
- Communicating with backend APIs over secure HTTP.
- Enforcing route-level access control.

The frontend does not store sensitive credentials or secrets. All configuration values related to backend communication are managed through environment variables at build time.

9.3 Backend Deployment

The backend API service is deployed as an independently hosted server application.

- **Deployment URL:** `https://iapss-backend.onrender.com/`
- **Hosting Type:** Managed cloud service

9.3.1 Backend Responsibilities in Deployment

- Handling authentication and request validation.
- Processing analysis requests.
- Enforcing access control and security policies.

The backend serves as the sole gateway to analysis services and persistent storage, ensuring that no internal services are directly exposed.

9.4 Environment Configuration

- Environment variables are used to configure API endpoints and service credentials.
- Sensitive configuration values are not hardcoded in the source code.
- Separate configurations can be maintained for development and production environments.

This approach reduces the risk of accidental credential exposure and supports secure deployment practices.

9.5 Communication Model

- All frontend–backend communication occurs over HTTPS.
- Requests are stateless and include required authentication headers.
- Cross-origin requests are restricted to trusted origins.

This model ensures controlled interaction between system components.

9.6 Security Implications of Deployment

The deployment architecture incorporates the following security considerations:

- Isolation of frontend and backend services reduces lateral attack vectors.
- Backend services are not directly accessible except through defined APIs.
- Token validation is enforced at the backend for all protected endpoints.

While the deployment is suitable for academic and controlled environments, it is not positioned as a hardened production-grade deployment.

9.7 Scalability and Availability

- Frontend hosting supports automatic scaling and global content distribution.
- Backend services can be scaled horizontally as demand increases.

The deployment model supports incremental growth without architectural changes.

This deployment architecture provides a balance between accessibility, security awareness, and operational simplicity, aligning with the project's intended scope and future extensibility.

Chapter 10

Limitations

This chapter outlines the known limitations of the Intelligent Academic Problem-Solving System (IAPSS). These limitations are documented to provide realistic expectations regarding the system's capabilities, scope, and operational boundaries.

The presence of these limitations does not indicate design flaws; rather, they reflect deliberate scoping decisions aligned with the project's intended purpose.

10.1 Scope Limitations

- The system focuses on analytical guidance and does not provide guaranteed correct or optimal solutions.
- Certain advanced features are intentionally deferred to future development phases.
- The platform is designed for academic and learning-oriented use cases rather than commercial or production-critical deployments.

10.2 Analysis Accuracy

- Analytical feedback is generated based on heuristic and model-driven interpretation.
- The system may not fully capture all edge cases or problem-specific nuances.
- Users are expected to exercise judgement and independent verification.

10.3 Security Constraints

- While security-aware design principles are followed, the system is not hardened against advanced persistent threats.
- Protection against large-scale denial-of-service attacks is outside the current scope.
- Security controls are appropriate for controlled and academic environments but may require enhancement for high-risk deployments.

10.4 Performance Constraints

- Response times may vary depending on external service availability and network conditions.
- High concurrency scenarios may lead to increased latency.

10.5 Dependency Constraints

- The system relies on third-party services for analysis and hosting.
- Availability and behaviour of external services are outside direct system control.

10.6 User Responsibility

- The system does not enforce correctness or academic compliance of user submissions.
- Users remain responsible for ethical and appropriate usage of the platform.

These limitations are documented to ensure transparency and to guide future improvements. Addressing selected limitations is considered part of the system's planned evolution.

Chapter 11

Future Scope (Phase-II Enhancements)

This chapter outlines potential enhancements and extensions planned for future versions of the Intelligent Academic Problem-Solving System (IAPSS). These features are not part of the current stable implementation and are documented separately to maintain clarity between existing functionality and planned evolution.

The future scope reflects both technical growth and expanded learning-oriented capabilities.

11.1 Advanced Analysis Capabilities

- Enhanced problem analysis with deeper contextual understanding.
- Adaptive feedback based on user skill level and interaction history.
- More granular optimisation and improvement suggestions for code analysis.

11.2 Competitive Programming Support

- Structured guidance tailored for competitive programming preparation.
- Identification of common patterns and pitfalls in algorithmic problems.
- Strategy-oriented hints focusing on time and space complexity improvement.

11.3 Extended Content Integration

- Integration of external informational sources such as curated news or updates related to technology and programming.
- Contextual recommendations for learning resources and practice material.

These integrations are planned cautiously to avoid information overload and to maintain the system's analytical focus.

11.4 User Personalisation

- Personalised dashboards reflecting user progress and interaction trends.
- Preference-based analysis presentation.
- Optional tracking of improvement over time.

11.5 Role-Based Access Control

- Introduction of administrative or moderator roles.
- Enhanced access control for managing system behaviour and content.

11.6 Security Enhancements

- Stronger monitoring and logging mechanisms.
- Rate limiting and abuse prevention strategies.
- Additional validation layers for sensitive endpoints.

11.7 Scalability and Deployment Improvements

- Improved handling of high concurrency scenarios.
- Deployment optimisations for increased availability.
- Support for alternative frontend clients.

These future enhancements are planned to be implemented incrementally, ensuring that each addition aligns with the system's core philosophy of analytical guidance, ethical usage, and maintainable design.

Chapter 12

Conclusion

The Intelligent Academic Problem-Solving System (IAPSS) represents a structured and security-aware approach to building an analytical assistance platform for academic and technical learning. The system is designed with a clear focus on helping users understand problems, improve reasoning, and refine coding practices rather than simply producing direct solutions.

Throughout this documentation, emphasis has been placed on strong architectural separation, modular design, and clearly defined system boundaries. The client–server model, stateless APIs, and controlled access mechanisms collectively contribute to a system that is maintainable, extensible, and resilient within its intended scope.

Security considerations have been treated as a foundational concern, with explicit enforcement of authentication, access control, and input validation. At the same time, the system maintains transparency regarding its limitations, ensuring realistic expectations and responsible usage. This balance between protection and openness reflects a deliberate and informed design philosophy.

The documentation clearly distinguishes between implemented functionality and planned enhancements, enabling future development without ambiguity or technical debt. By documenting both current capabilities and future scope, the system remains adaptable to evolving learning requirements, particularly in areas such as competitive programming support and personalised guidance.

In conclusion, IAPSS is positioned as a robust and thoughtfully engineered platform for analytical learning assistance. The architectural and security decisions documented herein provide a strong foundation for continued evolution, experimentation, and responsible innovation.

Appendix A

Appendix

This appendix contains supplementary information that supports the main documentation but is not part of the core system description. The materials included here are intended to aid future maintenance, onboarding, and reference.

A.1 Glossary

Term	Description
IAPSS	Intelligent Academic Problem-Solving System
API	Application Programming Interface
Authentication Token	A credential used to verify user identity for protected requests
Stateless API	An API design where each request contains all necessary context
Frontend	The client-side application responsible for user interaction
Backend	The server-side application handling logic and data processing

A.2 Abbreviations

- CP — Competitive Programming
- SRS — Software Requirements Specification
- UI — User Interface
- HTTP — Hypertext Transfer Protocol

A.3 Technology Stack Summary

- **Frontend:** React, Vite
- **Backend:** Node.js, REST APIs

- **Hosting (Frontend):** Vercel
- **Hosting (Backend):** Render
- **Authentication:** Token-based authentication

A.4 Version History

Version	Date	Description
v1.0	Initial Release	Core analysis, authentication, deployment

A.5 Notes for Future Maintainers

- Planned features are documented under the Future Scope chapter.
- Security-related changes should be reviewed against the Security Considerations chapter.
- Architectural changes should preserve stateless API principles.

A.6 Security Policy

This security policy defines the intended security posture, acceptable use, and responsible disclosure guidelines for the Intelligent Academic Problem-Solving System (IAPSS). The policy is provided to promote ethical usage, protect system integrity, and guide future development.

A.6.1 Purpose

The purpose of this security policy is to:

- Define acceptable and prohibited security-related activities.
- Encourage responsible interaction with the system.
- Document the scope and limitations of implemented security controls.

A.6.2 Intended Use

IAPSS is intended for:

- Academic learning and analytical assistance.
- Problem-solving guidance and code analysis.
- Controlled and ethical usage environments.

The system is not intended for commercial deployment, production-critical workloads, or handling sensitive personal data.

A.6.3 Supported Scope

Only the following are considered within supported scope:

- The actively deployed frontend and backend services.
- Documented API endpoints and authenticated user flows.

Experimental features, partially implemented modules, and future-scope integrations are excluded from supported scope.

A.6.4 Prohibited Activities

The following activities are strictly prohibited without prior authorisation:

- Attempting unauthorised access to backend services or databases.
- Exploiting API endpoints beyond documented usage.
- Performing penetration testing, vulnerability scanning, or automated attacks.
- Brute-force authentication attempts.
- Reverse engineering system behaviour for malicious intent.
- Deliberate service disruption or denial-of-service attempts.

Such activities may be treated as misuse of the system.

A.6.5 Input Handling and Data Safety

- User inputs are treated as untrusted by default.
- The system avoids execution of user-submitted content.
- Stored data is limited to what is necessary for system functionality.

Users are advised not to submit confidential or sensitive information.

A.6.6 Responsible Disclosure

If a security vulnerability or unintended behaviour is discovered:

- The issue should not be exploited.
- Public disclosure should be avoided.
- The issue should be reported responsibly to the project maintainer.

Reports should include a high-level description and reproduction steps without sensitive payloads.

A.6.7 Limitations and Disclaimer

While reasonable security practices are implemented:

- No guarantee is made against all forms of attack.
- The system is provided *as is*, without warranty.
- The author is not responsible for damages resulting from unauthorised use or misuse.

This policy is intended to promote ethical usage and transparent security expectations rather than claim absolute protection.

A.7 License

This document and the associated software are governed by a custom license intended to balance academic openness with protection of intellectual property.

A.7.1 License Type

The Intelligent Academic Problem-Solving System (IAPSS) and its documentation are released under a **Custom Academic and Reference Use License**.

A.7.2 Permitted Use

The following uses are permitted without prior written consent:

- Academic learning and personal study.
- Technical reference and documentation review.
- Evaluation for educational or research purposes.

A.7.3 Restricted Use

The following actions are not permitted without explicit authorisation:

- Commercial use or monetisation of the software or documentation.
- Redistribution of the source code or documentation in modified or unmodified form.
- Deployment of the system or its derivatives as a public service.
- Use of substantial portions of the system for derivative projects.

A.7.4 Attribution

Where reference to this project is made in academic or technical contexts, appropriate attribution to the original author is required.

A.7.5 No Warranty

The software and documentation are provided *"as is"*, without warranty of any kind, express or implied. The author makes no guarantees regarding correctness, reliability, or fitness for a particular purpose.

A.7.6 Limitation of Liability

In no event shall the author be held liable for any damages arising from the use, misuse, or inability to use the software or documentation.

A.7.7 License Enforcement

Violation of this license may result in:

- Revocation of usage permissions.
- Formal requests for takedown or removal.
- Reporting of misuse in academic or professional contexts.

This license is intended to protect academic originality while allowing responsible and ethical engagement with the system.

This appendix concludes the supporting documentation for IAPSS.