Bilkent University
Department of Computer Engineering

Senior Design Project
Team ID: T2521
Project Name: Ubien: The Immersive Inquiry Based Learning Platform

Detailed Design Report

Group Members:
22203367, Ahmet Deniz Gelir, deniz.gelir@ug.bilkent.edu.tr
22202254, Efe Can Tatar, can.tatar@ug.bilkent.edu.tr
22201820, Kemal Onur Özkan, onurozkan@ug.bilkent.edu.tr
22001736, Eren Uslu, eren.uslu@ug.bilkent.edu.tr
22203328,Simay Uygur, simay.uygur@ug.bilkent.edu.tr

Supervisor:

Uğur Güdükbay

Course Instructors:

İlker Burak Kurt

Mert Bıçakçı

13.03.2026

Contents

1. Introduction

1.1 Purpose of the system

It is commonplace in the present day to hear students bemoan that while they have “learned” a great deal on paper, they have retained only a fraction of the information they gathered over the course of their education. It is natural, then, to ask: what has been the point of learning all that information if it functioned merely as a temporary exercise in memorization? The current paradigms of education, from structured lectures to solution-driven problem sets and exam-oriented milestones, are built around the efficient delivery and assessment of knowledge rather than the cultivation of genuine understanding. In such systems, students often become passive recipients, carrying information only long enough to reproduce it when asked, and quickly losing it afterward.

This widespread pattern reveals a fundamental problem: the absence of active self-discovery. Educational psychology and centuries of pedagogical tradition alike affirm that concepts uncovered through one’s own reasoning are remembered more deeply, understood more clearly, and integrated more permanently into one’s intellectual framework [1]. Yet most modern educational tools unintentionally short-circuit this process. They present polished explanations, prepackaged insights, and step-by-step solutions that resolve uncertainty too quickly, preventing learners from engaging with the material in a way that forms resilient mental models.

Ubien arises from the recognition that true understanding occurs when students do not merely receive ideas, but are made to come up with it themselves. Drawing on the principles of Inquiry-Based Learning (IBL), Ubien transforms traditional IBL textbooks into interactive courses where students iteratively advance through answering a sequence of questions. Instead of being told what to think, learners are socratically questioned into discovering why concepts hold together and how the structures of a discipline emerge organically from first principles.

In this environment, knowledge is not an external artifact handed down by an instructor, it becomes something the student builds. The result is a form of understanding that is permanent, self-sourced, and meaningfully tied to the learner’s own cognitive efforts. Ubien seeks to restore this mode of discovery-based learning to the digital age, offering a platform where students engage with material in a way that mirrors the curiosity-driven process that underlies real intellectual growth.

As we proceed through this specification, we explore how Ubien leverages technology to revive the foundational experience of learning through inquiry, providing students with a modern tool that supports not only academic success but the development of independent, rigorous, and enduring understanding.

1.2 Design goals

1.2.1 Usability

- User interface should be easy to understand and free of seldom used clutter
- Users should be able to start or pick up a course they have with a single click
- Users should be able to upload textbook pdfs to create their own IBL courses without needing to switch tabs to verify their account and look at the created course.
- All core navigation actions, such as uploading textbooks, accepting cost estimates, selecting study modes, and asking questions, should be accessible in no more than four clicks from the dashboard.

1.2.2 Performance

- Under typical network conditions, text-based ChatGPT API responses will be delivered within three to five seconds.
- Text extraction and token estimation for uploaded PDF files up to 50 MB should be completed by the backend in 10 seconds or less.
- WebGL-based animations for the virtual instructor must maintain a vocal to animation delay less than 1 second.

1.2.3 Reliability

- The backend architecture will ensure an availability of at least 99% during regular operational hours.
- User progress, including completed tasks and questions asked, will be saved instantly to remain secure even if the browser is unexpectedly refreshed or closed.
- In the event of network failures, API timeouts, or invalid PDF uploads, the system will display descriptive error messages to the user within ten seconds and allow for request resubmission.

1.2.4 Scalability

- The backend should be capable of running multiple instances simultaneously to support up to 200 active users without any loss of performance.

- To reduce response times by at least 50% for repeated requests, frequently reused LLM outputs such as, textbook summaries and chapter metadata, will be cached.
- The backend code must be written according to SOLID principles to allow for easy functionality scalability.

1.2.5 Security

- Management of information security principles defined in ISO/IEC 27001 will be enforced for user authentication data, course data, and uploaded textbook materials.
- User data and conversational chat logs must remain confidential and encrypted.
- Users will be restricted from sharing their personalized courses or chat logs with others to protect private and copyrighted educational content.

1.3 Definitions, acronyms, and abbreviations

1.3.1 Definitions

- Ubien - Our AI assisted inquiry based learning platform, used as a hub for all the functionality that the service provides.
- Inquiry-Based Learning -An educational methodology where learners advance by answering a sequence of guided questions, discovering concepts through their own reasoning rather than passively receiving information.
- Virtual Instructor - An animated 2D/3D model capable of facial expressions, gestures, and synthesized speech that simulates the experience of a real educator to guide the user through the material.

1.3.2 Acronyms and abbreviations

- IBL: Inquiry based learning
- LLM: Large language model
- TTS: Text to speech
- UI: User interface

1.4 Overview

Modern educational paradigms often rely on the efficient delivery and assessment of knowledge, which can unintentionally reduce learning to passive memorization rather than active self-discovery. Ubien is designed to address this fundamental problem by reviving the experience of learning through inquiry. It is a digital education platform that transforms traditional Inquiry-Based Learning (IBL) textbooks into dynamic, self-guided online courses.

Powered by Large Language Models (LLMs) and a retrieval layer, Ubien ensures factual accuracy and an adaptive learning flow. To enhance user engagement, the platform integrates an animated virtual instructor, rendered via Unity WebGL, that is capable of speech synthesis, facial expressions, and gestures. Instead of simply providing prepackaged answers, Ubien uses Socratic questioning to guide students into discovering concepts themselves, resulting in a deeper and more permanent understanding of the material.

This Detailed Design Report provides a comprehensive technical roadmap for the development of Ubien. It details the system's three-layer architecture; comprising the Client, Backend, and Cloud layers, and outlines the specific technologies used to bring the platform to life. Furthermore, this document explores the functional and non-functional requirements , system constraints , ethical considerations, and test cases required to ensure that Ubien is delivered as a scalable, secure, and highly interactive educational tool.

2. Current software architecture

In the current market there are some education focused LLM products that aim to make education more accessible, cheaper and easier. Most of these solutions have functionality like turning textbooks into slide shows or questionnaires and smart recognition of text and summarizing capabilities. As Ubien we aim to offer a more learning centered approach to education rather than easier memorization. Still in a massively competitive market, there are some competitors that are similar to our product by nature. The most popular of these are YouLearn and Turbo AI.

YouLearn

YouLearn is an LLM based learning platform focusing on delivering an AI agent that can help students summarize textbooks, create quizzes and flashcards and answer questions. YouLearn also hosts a voice mode that can listen to your voice instead of needing text and it can read out what it is producing as well. This is the closest product to what we are making in the market.

The architecture of YouLearn seems to depend on AI agents to solve and summarize the problems in the backend with a user interactable frontend that is used for the visuals and interactions. This is the same system architecture that we plan to use for Ubien, so having a successful in-market

product with the same architecture works as a proof of concept for our product.
The key differences and advantages of Ubien over YouLearn are:

- Ubien isn't a summarization focused tool. While YouLearn's main functionality is being able to process text effectively and produce short outputs that can easily be summarized, Ubien aims to guide the user through the subject for a more in depth learning experience
- YouLearn lacks an instructor model that makes the application feel very information dense while using. Ubien will allow the user to switch to an instructor mode that will feel more comfortable and slower which will mimic the learning process of a classroom.
- While YouLearn is marketed towards students that need to memorize a lot and is used for cramming information, Ubien aims to market towards anyone that is interested in learning through discovery of a subject. The IBL approach is slower in nature but it produces better results for long term learning. This shifts the target audience from students that are studying for an exam to a broader audience that is genuinely interested in learning.

Turbo AI

Turbo AI is another LLM assisted learning platform that utilizes AI agents to create summaries and quizzes. Aside from the same functionality that the previous competitor YouLearn offered, Turbo AI also has functionalities like sharing notes and syncing web and mobile application data. This is highly convenient for users who frequently switch between devices or prefer collaborative studying, but it still fundamentally serves as a passive summarization and rote-memorization tool.

The key differences and advantages of Ubien over Turbo AI are:

- While Turbo AI excels at auto-generating quick quizzes and allowing users to share pre-made notes, it ultimately spoon-feeds information. Ubien, built strictly on IBL principles, actively challenges the user through Socratic questioning. It forces students to engage with the material and build their own mental models, ensuring deeper cognitive retention rather than temporary memorization.
- Turbo AI relies on standard text-based chats and traditional UI elements. Ubien differentiates itself by introducing an animated virtual instructor rendered in Unity WebGL. Complete with lip-syncing, facial expressions, and upper-body gestures, Ubien creates a more engaging, classroom-like environment that reduces study fatigue and keeps users focused.
- Turbo AI is heavily optimized for quick summaries of short texts or lecture notes. Ubien is specifically designed to handle full length, complex textbooks. By processing large PDFs, Ubien structures

comprehensive, multi-chapter courses rather than isolated study sessions.

3. Proposed software architecture

3.1 Overview

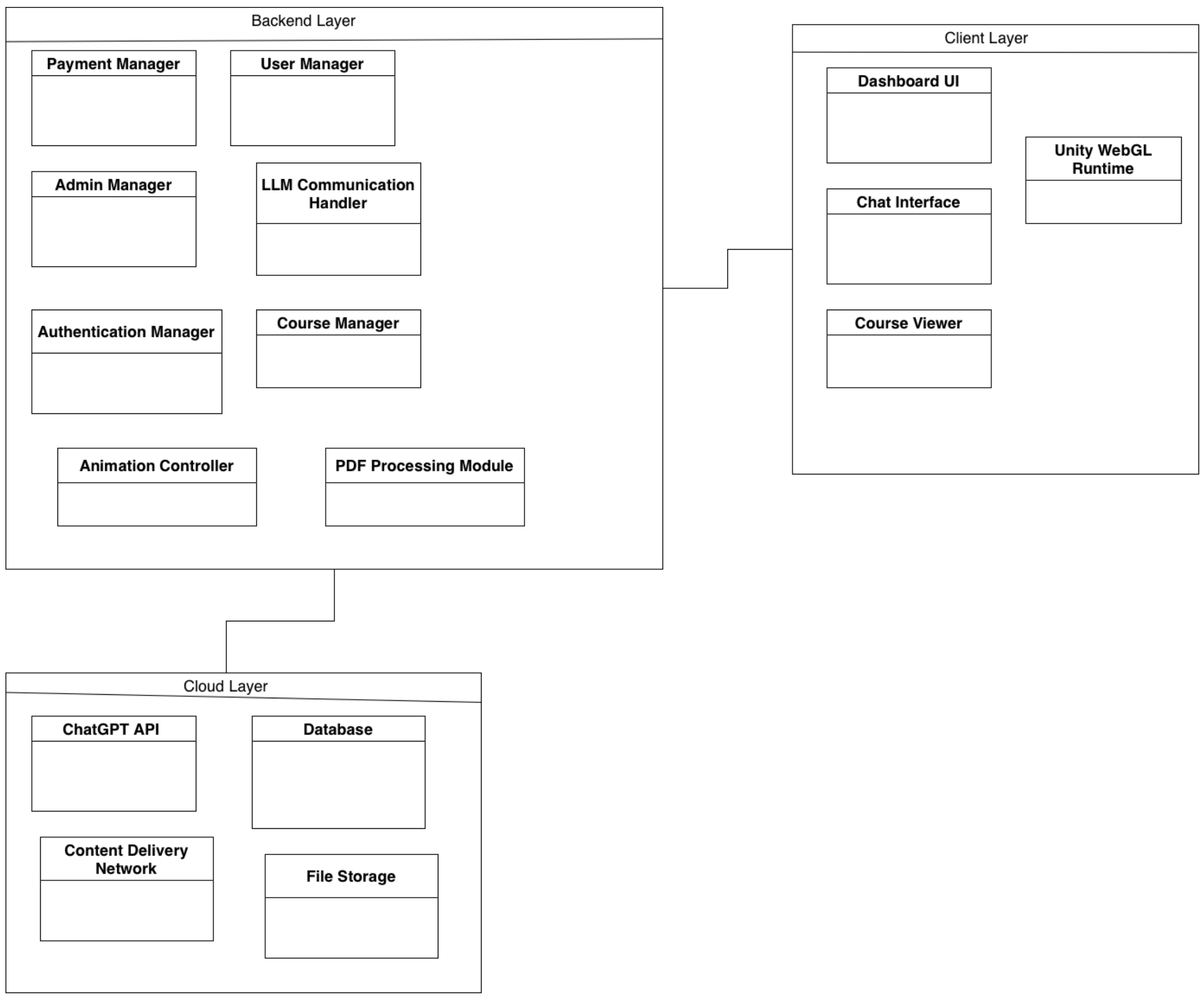

Figure 1: High Level System Architecture Diagram

The Client Layer, Backend Layer, and Cloud Layer make up the system's three-layer architecture. Together, these layers manage user contact, LLM-powered guidance, PDF submission, course creation, and animated teacher responses, as well as chatting with the course.

3.1.1 Client Layer

The client layer delivers the system's user interface. It utilizes HTML, CSS, and JavaScript, and operates entirely within the web browser. Users can engage with the AI instructor, explore their created courses, upload textbooks, and see animated model responses.

3.1.1.1 Dashboard UI

This component displays a summary of every course the user has created. It shows access buttons, creation dates, course titles, and progress status. It serves as the user's primary point of navigation.

3.1.1.2 Chat Interface

A natural language text field is provided for communicating with the virtual instructor through the Chat Interface. Users can participate in guided IBL discourse, ask questions about the course material, respond to the instructor's instructions, and request clarifications or repetitions as needed.

3.1.1.3 Course Viewer

Displays the generated IBL course content to the user, including question prompts, slides, progress indicators, and instructor interactions. It provides the main interface through which learners navigate the course, submit answers, and track their advancement.

3.1.1.4 Unity WebGL Runtime

The Unity WebGL instructor avatar is loaded and rendered by this component. It displays lip-syncing with TTS audio, eye contact, gestures, facial expressions, and upper-body movements.

3.1.2 Backend Layer

The system's primary business logic is managed by the backend layer. It generates all computational output provided to the client, manages user accounts, stores course data, processes uploaded PDFs, and interacts with external AI services.

3.1.2.1 Authentication Manager

All user authentication and security processes, including user registration, login/logout, email verification, and credential validation, are managed by this component. It guarantees that system functions are only accessible to verified users.

3.1.2.2 User Manager

User-related data, including profile details, preferences, and progress tracking, is managed by the User Manager. It controls how users engage with stored system data and their courses.

3.1.2.3 Admin Manager

For developers or maintainers, this module offers administrative features. Log viewing, system activity monitoring, error troubleshooting, and maintenance activities are some examples of what it could involve.

3.1.2.4 Course Manager

Generated course sections, IBL-style questions, and progress checkpoints are all stored and retrieved by the course manager. Gives users and the frontend clear access to all course data.

3.1.2.5 LLM Communication Handler

This component converts outputs into IBL course content, generates conversations for the Chat Interface, delivers PDF content, prompts, and questions to the ChatGPT API, and receives organized responses, serving as the primary entry point for the system's AI capabilities.

3.1.2.6 Animation Controller

It communicates with the Unity WebGL runtime via WebSockets to synchronize lip movements with generated audio and to apply gesture and emotion metadata received from the backend, ensuring expressive and coherent instructor animations.

3.1.2.7 PDF Processing Module

This module handles uploaded Inquiry-Based Learning (IBL) textbooks and prepares their content for use in the platform. It retrieves uploaded files from storage, extracts and preprocesses the text, and forwards the processed content to the LLM Communication Handler and Course Manager. This enables the system to transform static textbook documents into structured educational material.

3.1.3 Cloud Layer

The third-party services required for AI processing, multimedia rendering, and persistent storage comprise the cloud layer. These services supply the media structure and critical processing resources that the system needs.

3.1.3.1 ChatGPT API

All of the system's AI-powered functions are carried out by this component, including extracting and analyzing content from uploaded PDFs, constructing IBL-style courses, and generating explanations and guided inquiry stages. Additionally, it generates text-to-speech audio from instructor responses and facilitates natural-language conversation during study sessions.

3.1.3.2 Content Delivery Network

Large client-side content, including avatar models, textures, animation files, JavaScript bundles, and Unity assets, is efficiently stored and served by the CDN.

3.1.3.3 File Storage

File Storage is responsible for storing all user-uploaded materials. It manages the persistent storage of IBL textbooks (PDF files) and any auxiliary resources required by the platform.

3.1.3.4 Database

Database stores all persistent system-related information, including user accounts, course metadata, generated course materials, progress monitoring, token/cost history, chat history, and system logs.

3.2 Subsystem decomposition

Ubien is designed as a modular system in which the overall functionality is divided into several subsystems with clearly defined responsibilities. This decomposition allows the system to remain maintainable, scalable, and extensible while enabling individual subsystems to evolve independently. The main objective of this design is to separate concerns between the user interface, application logic, and external infrastructure services.

The system is organized into three major subsystems: Client, Backend, and Cloud. Each subsystem is responsible for a different layer of the application architecture and interacts with the others through well-defined interfaces. The Client subsystem manages user interaction and presentation, the Backend subsystem performs the main processing and coordination tasks, and the Cloud subsystem provides persistent storage and external AI services required by the platform.

Figure 2 illustrates the subsystem decomposition of the Ubien platform and the main interactions between its components.

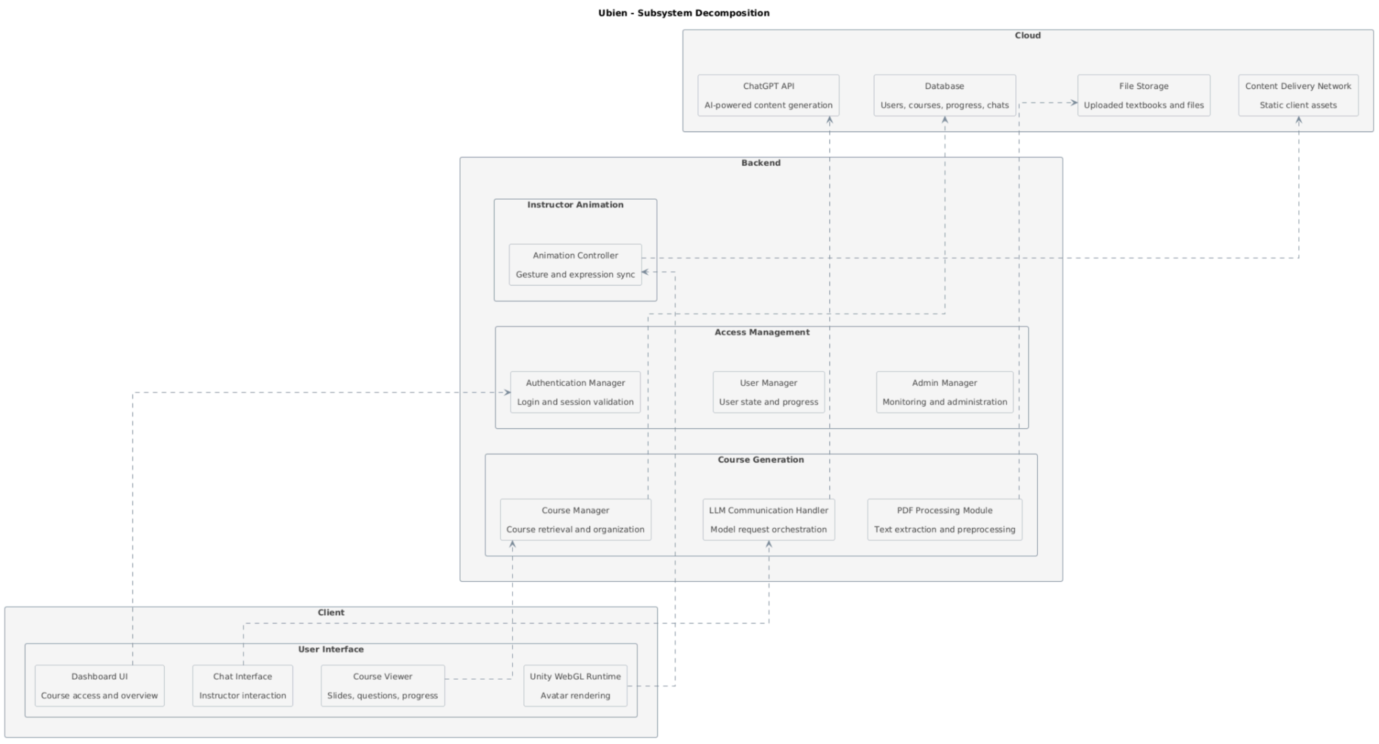

Figure 2: Ubien Subsystem Decomposition

The Client subsystem represents the user-facing portion of the platform and is responsible for presenting the learning environment to the user. All user interaction with the system occurs through this layer. Within the client subsystem, the User Interface module contains several components that together provide the main user experience. The Dashboard UI allows users to access available courses and view their learning progress. The Chat Interface enables conversational interaction between the learner and the AI-powered instructor, allowing users to ask questions and receive guidance during the learning process. The Course Viewer displays generated course material such as slides, exercises, and explanations in an organized format. In addition, the Unity WebGL Runtime renders the animated virtual instructor directly within the browser environment, allowing the instructor avatar to perform gestures and expressions synchronized with generated responses.

The Backend subsystem contains the core application logic of the system and coordinates communication between the client interface and the external services used by the platform. To improve modularity, backend functionality is organized into three functional groups: Access Management, Course Generation, and Instructor Animation.

The Access Management group is responsible for handling authentication, user state management, and administrative control of the system. The Authentication Manager manages login procedures and session validation. The User Manager maintains user-related information such as profiles, learning progress, and interaction history. The Admin Manager provides administrative monitoring capabilities that allow system administrators to supervise system activity and manage platform operations.

The Course Generation group is responsible for transforming uploaded textbook material into structured inquiry-based learning content. The PDF Processing Module extracts and preprocesses text from uploaded textbook files, preparing the data for further processing. The Course Manager organizes the generated material into a structured representation that can be presented within the learning interface. A central component of this group is the LLM Communication Handler, which manages all communication with the external language model service. This component constructs prompts, sends requests to the AI model, and processes the responses returned by the service.

The LLM Communication Handler is used both during the course generation process and during interactive conversations with the learner. When generating course content from textbook material, the module sends structured prompts to the language model in order to produce explanations, slides, and exercises. During interactive learning sessions, the same component processes messages from the Chat Interface and generates context-aware responses from the virtual instructor. By centralizing language model communication within a single module, the system ensures consistent prompt handling, simplifies integration with the AI service, and avoids duplication of functionality across different components.

The Instructor Animation group controls the behavior of the animated instructor displayed in the client interface. The Animation Controller synchronizes generated responses with the Unity WebGL runtime so that the instructor avatar can perform coordinated gestures and expressions while interacting with the learner. This component helps create a more engaging and immersive learning experience by visually representing the AI instructor during the learning process.

The Cloud subsystem provides external infrastructure services that support the operation of the platform. The ChatGPT API supplies the large language model capabilities used for both course generation and interactive instruction. The Database stores persistent information including user accounts, course structures, learning progress, and conversation histories. File Storage maintains uploaded textbook files and other large resources used during course generation. Finally, the Content Delivery Network (CDN) distributes static client-side assets such as WebGL builds and interface resources efficiently to end users.

This layered decomposition allows Ubien to maintain a clear separation between presentation logic, application processing, and infrastructure services. As a result, changes within one subsystem can typically be implemented without affecting the others. Such modularity improves system maintainability, supports scalability as the number of users grows, and enables future extensions such as new learning modalities, additional AI models, or alternative storage solutions.

3.3 Hardware/Software Mapping

Ubien is implemented as a **software-only** system and does not require any specialized hardware components. The platform runs entirely on standard cloud-based infrastructure and can be accessed through a modern web browser.
The client-side interface operates in the user’s browser and renders the dashboard, chat interface, and animated instructor using standard web technologies. The backend services are deployed on cloud servers and run inside Docker containers to ensure portability and reproducibility across development and production environments.
External services such as the OpenAI API and cloud storage systems provide additional computational capabilities required for AI-driven course generation, file management, and content delivery. Because all system components run on conventional server infrastructure, no dedicated hardware-software mapping is necessary beyond typical web server deployment.

3.4 Persistent data management

Persistent data in Ubien is managed through a PostgreSQL database and server-side file storage. The platform stores user-related information such as account details, created courses, learning progress, and interaction history in the database.
When a user uploads an Inquiry-Based Learning (IBL) textbook, the original PDF file is stored on the server as the canonical source of the

course content. The system processes this file to generate structured course data, including chapters, sections, slides, and questions, which are stored in the database and used to present course material to the user.
In addition, the system stores user progress information, such as completed slides and answered questions, allowing learners to resume their studies seamlessly. Chat interactions between the user and the AI instructor are also stored in the database to maintain conversational context during learning sessions.
Ubien does not require a specialized file system or custom database implementation. Instead, it relies on widely used database technologies and standard server storage mechanisms to ensure reliability, scalability, and maintainability.

3.5 Access control and security

Ubien uses a user authentication system to ensure that only registered users can access the platform. Users log in with their credentials, which are verified against records stored in the PostgreSQL database. After successful authentication, the server creates a session and assigns a session cookie to the client to maintain the user's authenticated state.
Access to system resources is restricted based on the authenticated user. Each user can only access their own courses, uploaded materials, learning progress, and interaction history. The backend verifies the user identity for each request and ensures that database queries are filtered using the corresponding user identifier.
Uploaded textbooks and generated course content are associated with the user who created them, preventing unauthorized access by other users. Communication between the client and server occurs through secure web protocols. For AI-powered features, selected processed content such as extracted textbook text or user queries may be sent to the external ChatGPT API to generate responses, while sensitive user data and authentication information remain within the platform.

4. Subsystem services

This section describes the services provided by each subsystem of the Ubien platform. These services define the responsibilities of the system components and the interactions between them. Each subsystem offers a set of services that support the overall operation of the platform while maintaining clear separation between presentation logic, application logic, and infrastructure services.

4.1 Client Subsystem Services

The Client subsystem is responsible for providing the graphical interface through which users interact with the Ubien platform. It presents learning content, collects user input, and communicates with the backend subsystem to request data and submit user actions. The client subsystem is implemented as a browser-based application and communicates with backend services through HTTP-based APIs and asynchronous requests.

The Dashboard UI provides services related to course navigation and user progress monitoring. It retrieves the list of available courses from the backend and displays progress indicators for each course. It also allows users to start new learning sessions or resume previously started courses.

The Chat Interface enables conversational interaction between the learner and the AI-powered instructor. This component captures user questions or responses and sends them to the backend subsystem. The backend then processes these inputs through the LLM Communication Handler and returns generated responses. The chat interface displays these responses within the conversation panel, allowing users to interact with the instructor in a natural dialogue format.

The Course Viewer is responsible for presenting generated learning materials to the user. It displays slides, explanations, and exercises produced during the course generation process. The course viewer retrieves structured course content from the backend subsystem and presents it in a sequential format that guides the learner through the material.

The Unity WebGL Runtime provides the visual rendering environment for the animated virtual instructor. This component loads the instructor avatar and receives animation signals from the backend. The animation controller synchronizes gestures, facial expressions, and other visual cues with generated instructor responses in order to enhance the learning experience.

4.2 Backend Subsystem Services

The Backend subsystem contains the core processing logic of the Ubien platform. It manages system state, coordinates communication between subsystems, and implements the main application functionality. Backend services are grouped into three functional areas: access management, course generation, and instructor animation.

The Authentication Manager provides user authentication and session management services. It validates login credentials, generates secure session tokens, and ensures that only authorized users can access protected system resources.

The User Manager provides services related to user data management. It stores user profiles, tracks course progress, and maintains interaction histories that allow users to resume learning sessions and maintain continuity across multiple sessions.

The Admin Manager supports system administration by providing monitoring and management services. Administrators can use this component to observe platform activity, review system status, and perform administrative actions when necessary.

The PDF Processing Module provides services for extracting and preprocessing textual content from uploaded textbook files. This module converts PDF documents into structured text that can be used as input for the course generation process.

The LLM Communication Handler provides the central interface for interacting with the external language model service. It constructs prompts, sends requests to the language model API, and processes the responses returned by the service. This component is used both during course generation and during interactive conversations between the learner and the AI instructor. By centralizing AI communication in a single component, the system ensures consistent prompt formatting, simplifies integration with the AI service, and prevents duplication of communication logic.

The Course Manager organizes the learning material produced by the system into structured courses. It receives processed textbook content and generated responses from the language model and converts them into structured elements such as slides, explanations, and exercises that can be displayed within the course viewer.

The Animation Controller manages synchronization between generated instructor responses and the visual representation of the instructor within the Unity WebGL runtime. It converts textual responses and metadata into animation signals that control gestures, facial expressions, and timing of the instructor avatar.

4.3 Cloud Subsystem Services

The Cloud subsystem provides external infrastructure services required for the operation of the platform. These services support data persistence, file storage, AI-powered processing, and efficient delivery of client-side resources.

The ChatGPT API provides the large language model capabilities used by the system. It generates learning content during course creation and produces responses to user questions during interactive learning sessions.

The Database provides persistent storage for system data. It stores user accounts, course metadata, progress records, conversation histories, and other information required for the operation of the platform.

File Storage stores uploaded textbook files and other large resources used during the course generation process. This service ensures that learning materials can be processed and reused across different learning sessions.

The Content Delivery Network (CDN) distributes static client-side resources such as interface assets, scripts, and WebGL builds. Using a CDN improves system performance by delivering these resources from geographically distributed servers, reducing load times for users.

5. Test Cases

5.1 Functional Test Cases

Test ID: 1
Category: Functional
Severity: Critical
Objective
Verify that a user can successfully register in the system.
Steps

1. Open the application in a web browser.
2. Click the Register button.
3. Enter valid email, username, and password.
4. Click Submit.

Expected Result
The system successfully creates a user account and sends a verification email.

Test ID: 2
Category: Functional
Severity: Major
Objective
Verify that users cannot log in with incorrect credentials.
Steps

1. Open the login page.
2. Enter a valid email and an incorrect password.
3. Click Login.

Expected Result
The system displays an Invalid credentials message and denies access.

Test ID: 3
Category: Functional
Severity: Critical
Objective
Verify that a verified user can successfully log in.
Steps

1. Navigate to the login page.
2. Enter valid credentials.
3. Click Login.

Expected Result
The user is redirected to the Dashboard page.

Test ID: 4
Category: Functional
Severity: Major
Objective
Verify that users can successfully log out.
Steps

1. Log into the system.
2. Click the Logout button.

Expected Result
The system logs the user out and redirects to the login page.

Test ID: 5
Category: Functional
Severity: Major
Objective
Verify that the dashboard displays all user courses.
Steps

1. Login to the system.
2. Navigate to the Dashboard page.

Expected Result
All previously created courses appear with title, creation date, and progress.

Test ID: 6
Category: Functional
Severity: Major
Objective

Verify that a user can create a new course.
Steps

1. Log into the system.
2. Navigate to the Dashboard.
3. Click Create Course.
4. Upload a valid PDF textbook.
5. Submit the form.

Expected Result
The system successfully creates a new course entry.

Test ID: 7
Category: Functional
Severity: Minor
Objective
Verify that the course progress status is displayed correctly.
Steps

1. Log into the system.
2. Navigate to the Dashboard.
3. Observe the progress indicator for each course.

Expected Result
Each course shows correct progress information.

Test ID: 8
Category:Functional
Severity: Critical
Objective
Verify that users can upload a valid PDF file.
Steps

1. Log into the system.
2. Click Create Course.

3. Upload a valid PDF file.
4. Click Submit.

Expected Result
The PDF file is uploaded and processed by the backend.

Test ID: 9
Category: Functional
Severity: Major
Objective
Ensure the system rejects unsupported file types.
Steps

1. Log into the system.
2. Click Create Course.
3. Attempt to upload a .docx file.

Expected Result
The system rejects the upload and shows an error message.

Test ID: 10
Category: Functional
Severity: Major
Objective
Verify that the system extracts text from the uploaded PDF.
Steps

1. Upload a valid PDF file.
2. Wait for the processing to complete.

Expected Result
Text is successfully extracted and sent to the AI service.

Test ID: 11
Category: Functional
Severity: Critical
Objective

Verify that the system generates an IBL course from a PDF.
Steps

1. Upload a valid textbook PDF.
2. Wait for course generation.
3. Open the Course Viewer.

Expected Result
The system generates structured course content.

Test ID: 12
Category: Functional
Severity: Major
Objective
Verify that the AI generates meaningful questions from course content.
Steps

1. Generate a course from a PDF.
2. Navigate to the Course Viewer.

Expected Result
The system displays meaningful questions based on the course material.

Test ID: 13
Category: Functional
Severity: Major
Objective
Verify system behavior if AI fails to generate course content.
Steps

1. Upload a PDF with minimal or unreadable text.
2. Start course generation.

Expected Result
The system shows an appropriate error message.

Test ID: 14
Category: Functional
Severity: Major
Objective
Verify that users can send messages to the AI instructor.
Steps

1. Open the Chat Interface.
2. Type a question.
3. Click Send.

Expected Result
The AI instructor returns a relevant response.

Test ID: 15
Category: Functional
Severity: Major
Objective
Verify that chat history is preserved during a session.
Steps

1. Send multiple messages in the chat interface.
2. Scroll through the chat history.

Expected Result
Previous messages remain visible.

Test ID: 16
Category: Functional
Severity: Major
Objective
Verify that users can request clarification from the AI instructor.
Steps

1. Ask a question in the chat.
2. Ask a follow-up question.

Expected Result
The AI responds contextually based on the previous message.

Test ID: 17
Category: Functional
Severity: Major
Objective
Verify that generated course slides are displayed correctly.
Steps

1. Open a generated course.
2. Navigate through course sections.

Expected Result
Slides display correctly.

Test ID: 18
Category: Functional
Severity: Major
Objective
Verify that users can answer course questions.
Steps

1. Open a course section.
2. Select an answer to a question.

Expected Result
The system records the answer.

Test ID: 19
Category: Functional
Severity: Major
Objective
Verify that course progress is updated after answering questions.
Steps

1. Complete a question in the course viewer.

Expected Result
Progress indicator updates accordingly.

Test ID: 20
Category: Functional
Severity: Major
Objective
Verify that the instructor avatar appears in the interface.
Steps

1. Open the Course Viewer.
2. Observe the instructor avatar.

Expected Result
The avatar loads successfully.

Test ID: 21
Category: Functional
Severity: Major
Objective
Verify that the avatar lip-syncs with generated speech.
Steps

1. Ask a question to the AI instructor.
2. Wait for the spoken response.

Expected Result
Lip movements match the speech.

Test ID: 22
Category: Functional
Severity: Major
Objective
Verify that the avatar performs gestures during explanations.
Steps

1. Ask a question that triggers a long explanation.

Expected Result
The avatar performs gestures and expressions.

Test ID: 23
Category: Functional
Severity: Major

Objective
Verify profile information can be updated successfully.
Steps

1. Navigate to Profile Page.
2. Click “Edit”.
3. Change profile details.
4. Click “Save”

Expected Result
Profile is updated and a success message appears.

Test ID: 24
Category: Functional
Severity: Major
Objective
Verify error is shown when profile update fields are left empty.
Steps

1. Navigate to Profile Page.
2. Click “Edit”.
3. Leave all fields empty.
4. Click “Save”.

Expected Result
Profile is not updated and an error message appears.

Test ID: 25
Category: Functional
Severity: Major

Objective
Verify that a user can successfully subscribe to a plan.

Steps

1. Navigate to Subscription page.

2. Select a subscription plan.
3. Enter payment details.
4. Confirm subscription.

Expected Result
The user’s subscription is activated and reflected in their account.

Test ID: 26
Category: Functional
Severity: Major

Objective
Verify that subscription fails when payment is declined.

Steps

1. Navigate to Subscription page.
2. Select a subscription plan.
3. Enter invalid or declined payment details.
4. Confirm subscription.

Expected Result
Subscription is not activated and an error message is displayed.

Test ID: 27
Category: Functional
Severity: Critical
Objective
Verify AI-generated question creation.
Steps

1. Navigate to Course Creation page.
2. Upload source material.
3. Click “Generate Course”.

Expected Result
AI-generated questions is created and displayed successfully.

Test ID: 28
Category: Functional
Severity: Major

Objective
Verify that user progress is saved correctly in the database.

Steps

1. Open a course and answer several questions.
2. Log out.
3. Log in again and reopen the same course section.

Expected Result
All previously completed questions and slide positions are restored correctly.

Test ID: 29
Category: Functional
Severity: Major

Objective
Verify that course creation is blocked when monthly quota is exhausted.

Steps

1. Navigate to Course Creation page.
2. Attempt to create a new course with quota exceeded.

Expected Result
Course creation is blocked and an appropriate warning is displayed.

Test ID: 30
Category: Functional
Severity: Critical
Objective
Verify AI answers adapt to user-provided context.
Steps

1. Provide context or background info to AI.
2. Ask a related question.

Expected Result
AI response reflects the provided context accurately.

Test ID: 31
Category: Functional
Severity: Major

Objective
Verify that the user can access course content.

Steps

1. Navigate to the Dashboard.
2. Select a course.
3. Open a chapter and section.

Expected Result
The selected course content is displayed correctly.

Test ID: 32
Category: Functional
Severity: Major

Objective
Verify that a user can navigate slides within a section.

Steps

1. Open a course section with slides.
2. Navigate forward and backward between slides.

Expected Result
Slides are displayed correctly and progress is tracked.

Test ID: 33
Category: Functional
Severity: Major
Objective
Verify that progress is correctly updated when a user completes a section.
Steps

1. Complete all slides and questions in a section.
2. Check the progress view.

Expected Result
The completed section is marked correctly in the progress dashboard.

Test ID:34
Category: Functional
Severity: Major
Objective
Verify that a user can successfully change the language in settings.
Steps

1. Open account settings.
2. Change the language from the dropdown menu.

Expected Result
The app interface updates to the selected language immediately.

Test ID: 35
Category: Functional
Severity: Major
Objective

Verify that a user’s course creation quota is not decremented if course generation fails.
Steps

1. Start creating a course.
2. Simulate failure in the AI service during generation.
3. Check the user’s remaining course quota.
   Expected Result
   The monthly course creation quota remains unchanged if course generation fails.

5.2 Non-Functional Test Cases

Test ID: 36
Category: Performance
Severity: Critical
Objective
Verify that AI responses are generated within acceptable time.
Steps

1. Ask a question in the chat interface.
2. Measure response time.

   Expected Result
   The response is generated within 5 seconds.

Test ID: 37
Category: Performance
Severity: Major
Objective
Verify system performance during multiple simultaneous users.
Steps

1. Simulate 50 users interacting with the system simultaneously.

Expected Result
System performance remains stable.

Test ID: 38
Category: Security

Severity: Critical
Objective
Verify that unauthorized users cannot access course data.
Steps
1. Attempt to access another user's course via URL.

Expected Result
Access is denied.

Test ID: 39
Category: Security
Severity: Critical
Objective
Verify that user passwords are securely stored.
Steps
1. Inspect database entries.

Expected Result
Passwords are stored in encrypted/hashed format.

Test ID: 40
Category: Reliability
Severity: Major
Objective
Verify system stability if AI service is temporarily unavailable.
Steps
1. Simulate AI API failure.

Expected Result
System displays a proper error message.

Test ID: 41
Category: Compatibility
Severity: Major
Objective
Verify cross-browser compatibility.
Steps
1. Run the application in Chrome, Firefox, Safari, and Edge.

Expected Result
Application functions correctly in all browsers.

Test ID: 42
Category: Usability
Severity: Minor
Objective

Verify that the interface is understandable for new users.
Steps

1. Conduct a usability test with new users.

Expected Result
Users can complete basic tasks without assistance.

Test ID: 43
Category: Reliability
Severity: Major
Objective
Verify that the system saves course progress correctly.
Steps

1. Answer several course questions.
2. Log out and log in again.

Expected Result
Progress is preserved.

Test ID: 44
Category: Performance
Severity: Minor
Objective
Verify login operation completes in less than 2 seconds.
Steps

1. Enter credentials.
2. Click “Sign In”.

Expected Result
Login operation completes in <2 seconds.

Test ID: 45
Category: Performance
Severity: Minor
Objective
Verify AI course generation completes in less than 10 minutes.
Steps

1. Upload PDF.
2. Request AI course generation.

Expected Result
Generated course appears in <10 minutes.

Test ID: 46
Category: Performance
Severity: Minor
Objective
Verify loading indicator appears while fetching AI course data.
Steps

1. Navigate to course generation page.

Expected Result
Loading icon is shown while fetching data .

Test ID: 47
Category: Performance
Severity: Minor
Objective
Verify course interface loads within 5 seconds.
Steps

1. Navigate to courses
2. Select a course.

Expected Result
Page fully loads and ready in ≤5 seconds.

Test ID: 48
Category: Security
Severity: Major
Objective
Verify that unauthorized users cannot access subscription-protected courses.
Steps

1. Attempt to open a course without an active subscription.

Expected Result
The system blocks access and displays a subscription-required message.

Test ID: 49
Category: Reliability
Severity: Major
Objective
Verify that the Ubien web app is accessible through the official website.
Steps

1. Open a web browser.
2. Navigate to the Ubien web application URL.
3. Attempt to log in or access the dashboard.

Expected Result
The web app loads successfully and is fully accessible to authorized users.

Test ID: 50
Category: Compatibility
Severity: Major
Objective
Verify that the Ubien web app interface scales correctly on different device screen sizes.
Steps

1. Open the Ubien web app on devices of various screen sizes (desktop, tablet, mobile).
2. Navigate through key pages such as dashboard, course view, and settings.
3. Check for visibility and accessibility of all text and controls.

Expected Result
The UI elements are correctly scaled and fully visible on all tested devices without overlap or cut-off text.

6. Consideration of Various Factors in Engineering Design

6.1 Constraints

6.1.1. Implementation Constraints

- All course generation and TTS must use the OpenAI API through the backend.
- The server must be implemented in Go, with WebSockets for live interaction.
- User accounts, courses, and progress must be stored in PostgreSQL.
- User login must rely on secure server-side sessions (cookie-based), also required for WebSocket upgrades.
- The system must run in Docker containers, both for development and deployment.
- The project must use GitHub for code, issues, and basic CI.

6.1.2. Economic Constraints

- API costs and token restrictions may limit the number of interactions that a user can make in a month of subscription.
- The API costs will be covered from the funds obtained from monthly subscriptions. Users will be allowed to use as much as they pay.
- All software will be built using freely available technologies.
- For an initial setup intended to serve up to 100 users, the monthly hosting costs amount to approximately 45 $, broken down as follows: 15 $ (backend server) + 15 $ (managed PostgreSQL DB) + 5 $ (storage) + 10 $ (bandwidth) + 1 $ (domain/DNS) + 0 $ (SSL) + 0 $ (CDN, Cloudflare Free Tier). In the long run, these hosting costs will be covered by allocating a portion of the revenue obtained from monthly subscriptions, allowing Ubien to operate sustainably and profitably.

6.1.3. Ethical Constraints

- User data and their chats must stay confidential and encrypted
- Users won’t be allowed to share chat logs and courses with others as the course may include personalized or copyrighted material for the user.
- All datasets and educational resources used will comply with fair-use, privacy, and ethical data-handling principles
- All visual and instructional material must be correctly attributed and comply with fair-use principles.

6.2 Standards

- The UML 2.5.1 standard (ISO/IEC 19505) will be followed in all diagrams, to ensure uniform, formal, and easily recognizable documentation [2].

- The IEEE 830 Software Requirements Specification (SRS) standard is followed when writing both functional and non-functional requirements to guarantee that they are unambiguous, traceable, measurable, and precise [3].
- Management of information security principles defined in ISO/IEC 27001 will be used for user authentication data, course data, and uploaded textbook materials [4]. This covers risk-aware system behavior, restricted access, and safe storage procedures.
- To maintain professional and consistent project reports, the IEEE referencing style will be used for all academic citations and references [5].
- To ensure the proper processing of model outputs, a compliant request structure, and the responsible use of LLM capabilities, all AI-powered interactions (LLM-based course creation, PDF reading, and TTS) must adhere to OpenAI's API usage regulations [6].
- For ethical conduct in the system's design, development, and delivery, all developers will adhere to the ACM Code of Ethics and Professional Conduct [7]. This involves accountability for user privacy, openness, equity, and acknowledgment of the project's educational goal.
- WebGL will be used to run the animated teacher, adhering to W3C standards for browser-based graphics for rendering effectiveness and web browser compatibility [8].
- The system will be implemented in accordance with Clean Code standards to ensure readability, maintainability, and continuous extensibility [9].

7. Teamwork Details

7.1 Contributing and functioning effectively on the team

Each team member contributed according to their technical expertise and assigned responsibilities.

Ahmet Deniz Gelir contributed to both backend and frontend development. Ahmet Deniz created the main structure of the project in the Go language, which is responsible for generating courses from IBL coursebooks. He is working on optimizing the course creation of the project, working on issues such as the context memory of the LLMs.

Simay Uygur works mainly on backend development. Simay contributed to the design and implementation of backend services, improving backend flow and identifying and fixing issues such as to LLM interactions with PDFs.

Eren Uslu was responsible for testing activities. Eren focused on verifying system functionality, identifying potential bugs, and ensuring that the system met functional and performance requirements.

Kemal Onur Özkan is primarily responsible for developing the Unity-based animations and interactive components of the system. This included implementing the visual interface, user interactions, and integration between Unity and backend services.

Efe Can Tatar is contributing to the Unity development tasks. He works to create movement animations of the Unity models, optimizes the interactive environment, and improves the visual and functional aspects of the application.

7.2 Helping creating a collaborative and inclusive environment

The team maintained a collaborative working environment throughout the project. Team members communicated regularly to discuss design decisions, implementation progress, and potential issues.

Frequent discussions and feedback sessions allowed team members to share ideas and improve system design collectively. All members were encouraged to contribute suggestions and participate in decision-making processes.

Additionally, task distribution was organized in a way that allowed team members to support each other when necessary. If one part of the system encountered difficulties, other members assisted in debugging or problem solving. This collaborative approach helped maintain steady progress and strengthened team cohesion.

7.3 Taking lead role and sharing leadership on the team

Leadership responsibilities in the project were distributed among team members according to their areas of expertise. Instead of having a single centralized leader, the team adopted a distributed leadership approach, where different members took the lead in specific technical domains. This approach allowed each subsystem of the project to be developed efficiently while ensuring coordination between different components.

Ahmet Deniz Gelir took the lead role in the core backend architecture of the system. He designed the main structure of the project in the Go programming language and implemented the foundational components responsible for generating courses from IBL coursebooks.

Simay Uygur assumed a leadership role in backend service reliability and system flow management. She focused on improving the interaction between backend services and large language models.

Eren Uslu led the testing and quality assurance efforts of the project. He was responsible for verifying system functionality, identifying potential issues, and ensuring that the application met its functional and performance requirements.

Kemal Onur Özkan led the development of the Unity-based interactive interface and animation systems. He guided the design and implementation of the visual components of the platform, including the interactive elements that allow users to engage with the system.

Efe Can Tatar took leadership in the development of character movement and animation systems within the Unity environment. He worked on creating and optimizing movement animations for Unity models, improving the realism and responsiveness of the interactive environment.

By distributing leadership responsibilities across different technical areas such as backend architecture, backend service reliability, frontend interaction design, animation systems, and system testing, the team ensured balanced participation and effective coordination throughout the development process.

8. Glossary

API:	Application Programming Interface
LLM:	Large Language Model
CDN:	Content Delivery Network

9. References

[1] I. Kaiser, J. Mayer, and D. Malai, “Self-Generation in the Context of Inquiry-Based Learning,” *Frontiers in Psychology*, vol. 9, 2018. doi: 10.3389/fpsyg.2018.02440.

[2] “About the Unified Modeling Language Specification Version 2.5.1,” *www.omg.org*. https://www.omg.org/spec/UML/2.5.1/

[3] “IEEE SA - IEEE 830-1998,” *IEEE Standards Association*. https://standards.ieee.org/ieee/830/1222/ (accessed Nov. 20, 2025).

[4] ISO, “ISO/IEC 27001,” *ISO*, 2022. https://www.iso.org/standard/82875.html (accessed Nov. 20, 2025).

[5] “IEEE Editorial Style Manual,” *IEEE Author Center Journals*. https://journals.ieeeauthorcenter.ieee.org/create-your-ieee-journal-article/create-the-text-of-your-article/ieee-editorial-style-manual/ (accessed Nov. 20, 2025).

[6] OpenAI, “Usage policies,” *Openai.com*, 2024. https://openai.com/policies/usage-policies/ (accessed Nov. 20, 2025).

[7] Association for Computing Machinery, “ACM Code of Ethics and Professional Conduct,” *Association for Computing Machinery*, Jun. 22, 2018. https://www.acm.org/code-of-ethics (accessed Nov. 20, 2025).

[8] W3C, “Standards - W3C,” *W3.org*, 2019. https://www.w3.org/standards/ (accessed Nov. 20, 2025).

[9] R. C. Martin, *Clean Code a Handbook of Agile Software craftmanship.* Prentice Hall, 2008.