Sprint Report - Coding Learning Platform (G-17)

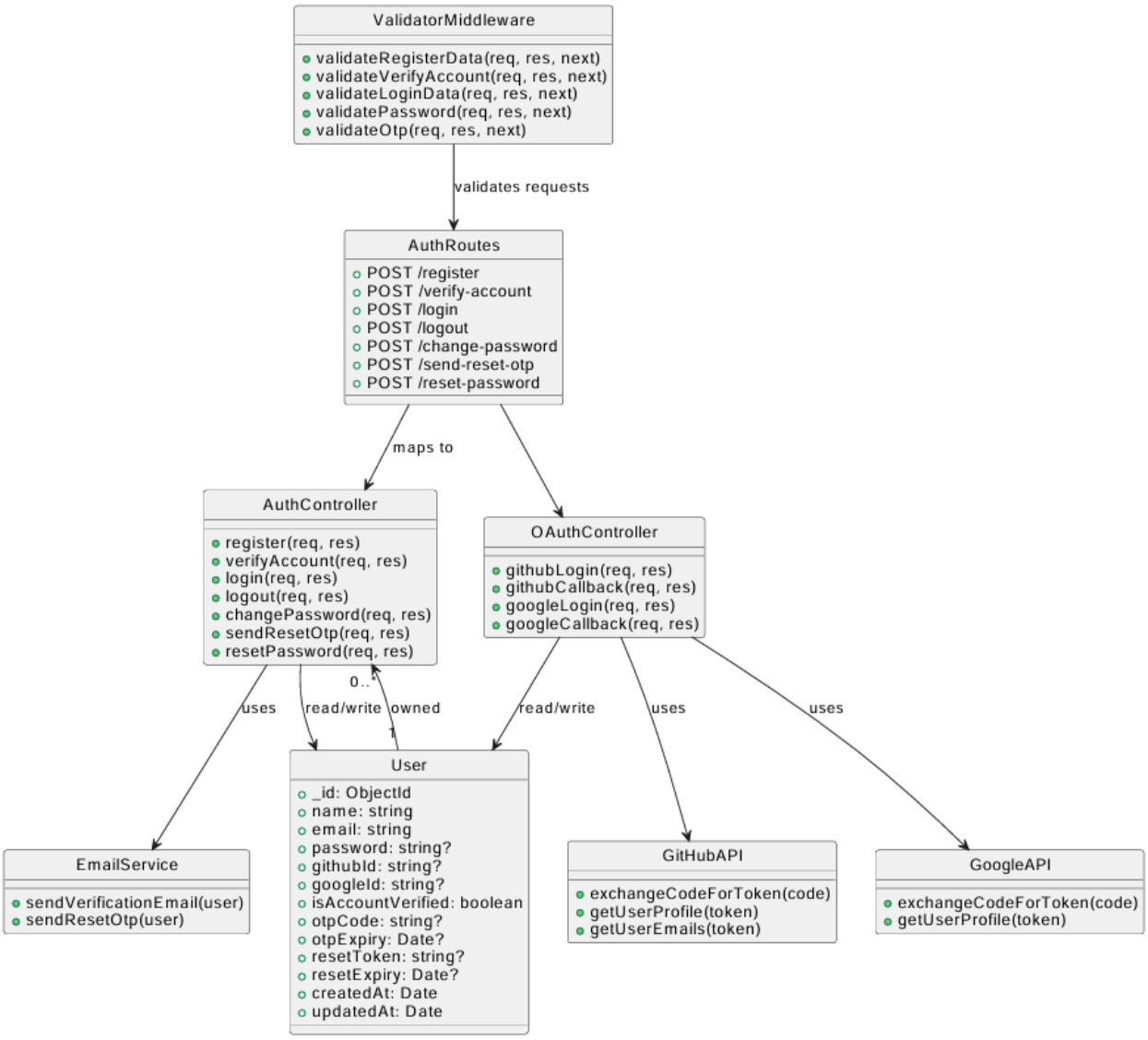
Sprint 1:

Sprint Goal: Implement a complete authentication system for the Coding Learning Platform.

Work done:

- User registration with validation.
- Secure login using JWT.
- Password hashing with bcrypt.
- Integrated Google OAuth 2.0.
- Auto-create users and verify tokens.
- Implemented GitHub OAuth 2.0.
- Linked GitHub accounts to platform profiles.
- Verification email setup.
- Secure verification token + verification route.
- Login restricted until verified.
- Generated secure reset token.
- Implemented reset password flow.

Class Diagram:



Work left: none, everything that was decided has been implemented

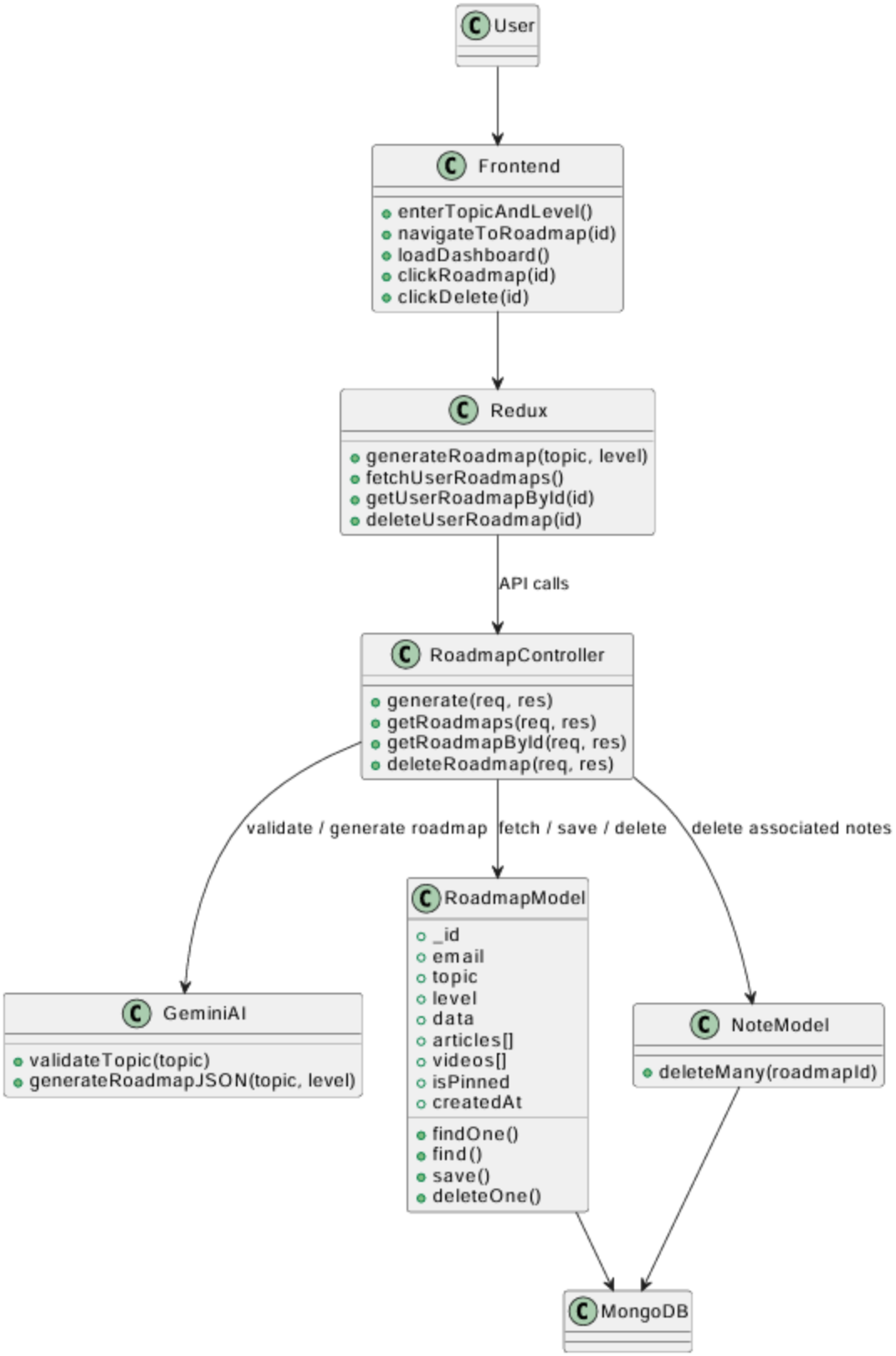
SPRINT 2:

Sprint goal: Implement an integrated roadmap generation module

Work done:

- Implemented topic and level input flow on the frontend for roadmap creation.
- Added dashboard loading, roadmap viewing, and delete interactions on the UI.
- Integrated Redux actions for generating a roadmap, fetching user roadmaps, retrieving a roadmap by ID, and deleting a roadmap.
- Built backend RoadmapController with full support for generate, getRoadmaps, getRoadmapById, and deleteRoadmap.
- Connected Gemini AI to validate topics and generate the roadmap JSON structure.
- Implemented the RoadmapModel with fields for email, topic, level, data, articles, videos, pinned status, and timestamps.
- Completed database operations, including saving new roadmaps, retrieving all roadmaps for a user, fetching details by ID, and deleting specific roadmaps.
- Implemented automatic deletion of associated notes through the NoteModel when a roadmap is removed.
- Established a complete end-to-end flow from frontend -> Redux -> API -> AI -> database.

Class diagram:



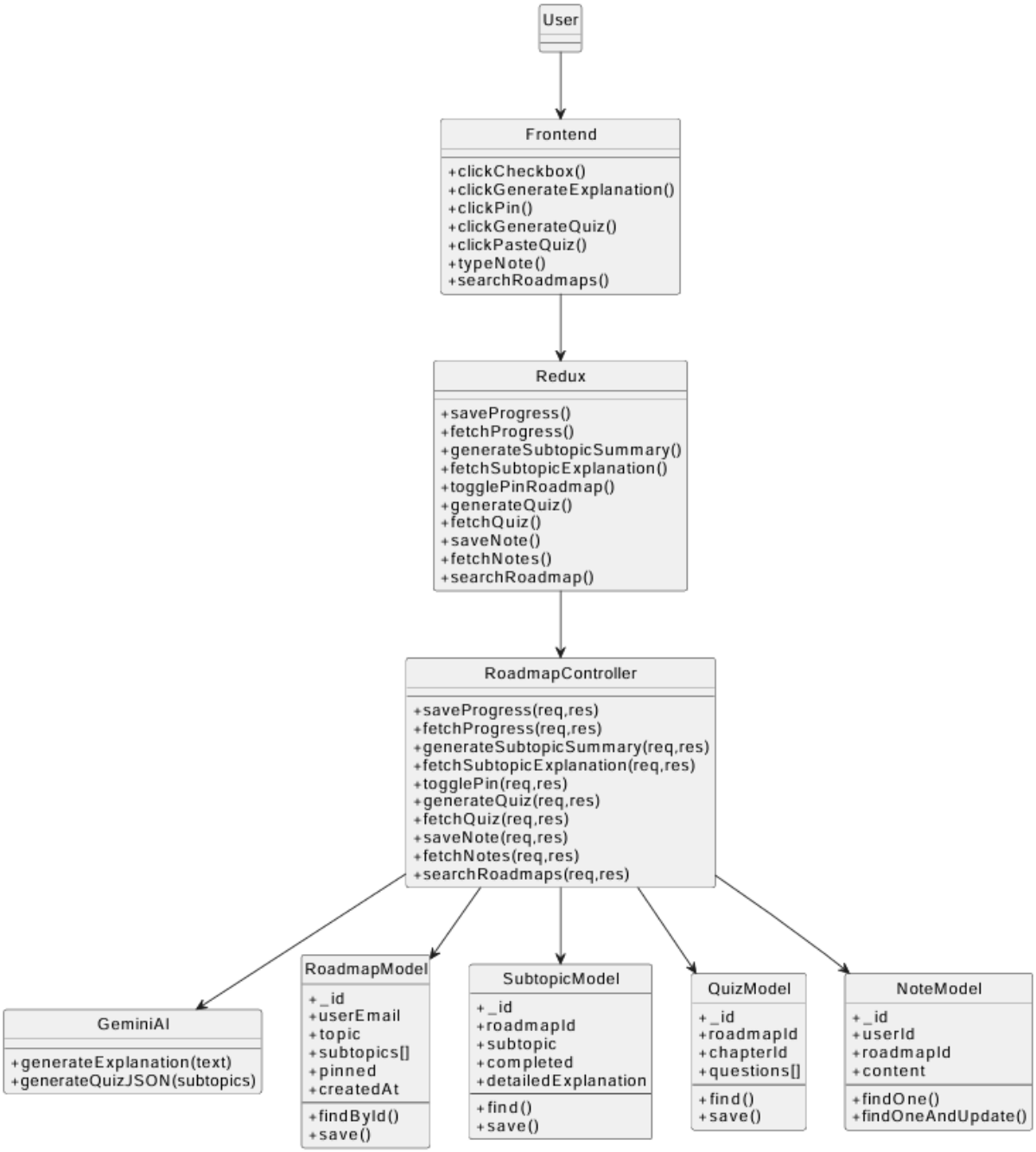
Work left: none, everything was completed

SPRINT 3:

Sprint goal: Implement features like quiz, summary, videos and articles fetching, notes taking

Work done:

- Added roadmap interaction features: marking progress, generating explanations, generating quizzes, pasting quizzes, adding notes, pinning roadmaps, and searching roadmaps.
- Implemented fetching of YouTube video links and relevant articles for each subtopic as part of learning resources.
- Added Redux actions and API routes for progress save/fetch, subtopic summaries, detailed explanations, quizzes, notes, roadmap search, and resource fetching.
- Integrated Gemini AI for generating explanations and quiz data.
- Updated RoadmapModel and SubtopicModel to store subtopics, explanations, videos, articles, and pinned status.
- Added QuizModel for storing generated questions and NoteModel for user notes.



Work left: none, everything was completed

SPRINT 4:

Sprint goal: Implement IDE for code practice and Chat-bot for help and guidance

Work done:

- Developed a fully functional in-browser IDE with support for all major programming languages.
- Implemented automatic language detection based on the user's code input.
- Added real-time code execution, output display, and error handling within the IDE environment.
- Integrated an AI chatbot directly into the coding interface to provide guidance, explanations, debugging help, and roadmap-related support.
- Connected the chatbot with user context so it can answer questions based on the active roadmap or coding task.


User
interacts
ChatBox
+ sendMessage(message)
+ appendMessage(msg)
+ clickRename()
+ clickDelete()
dispatches actions
ChatSlicer
+ createChat(payload)
+ getChatResponse(chatId, message)
+ fetchChats(roadmapId, moduleId)
+ getChat(chatId)
+ renameChat(chatId, newTitle)
+ deleteChat(chatId)
calls API endpoints
ChatController
+ createChat(req,res)
+ getResponse(req,res)
+ getChats(req,res)
+ getChat(req,res)
+ renameChat(req,res)
+ deleteChat(req,res)
generates title / responses
read/write chat documents
GeminiAI
+ generateTitlePrompt()
+ generateResponsePrompt()
+ generateResponse(input)
Represents the LLM system
(Gemini / AI model)
ChatModel
+ _id
+ title
+ messages
+ userEmail
+ roadmapId
+ moduleId
+ createdAt
+ updatedAt
+ findOne(query)
+ findById(chatId)
+ findByIdAndDelete(chatId)
+ save()
persists data
MongoDB
Mongo-like schema for chat storage

Work left: none, everything was completed

SPRINT 5:

Sprint goal: Implement Code controller, add time and space complexity analysis, and run code in real time

Work done:

- Implemented the CodeController to handle code input, execution requests, and response handling.
- Added real-time code execution with support for multiple programming languages.
- Implemented backend logic to compute and return time and space complexity estimations for submitted code.
- Added UI support to display execution output, errors, and complexity analysis results.
- Ensured smooth integration between the IDE, code execution engine, and analysis module.