Assignment 4 – Software Architecture

Group: 2301004, 2301045, 2301086

I. Selected Software Architecture Style

Chosen Architecture:

Model–View–Controller (MVC) Architecture implemented using Node.js (Express + Mongoose)

A. Justification of Architecture Style and Component Granularity (5 Marks)

Model Layer

- Implemented using Mongoose schemas.
- Stores:
  - User credentials (hashed passwords)
  - Steam profile information.
- Responsible only for database operations.
- No business or routing logic included.

View Layer

- Web-based chatbot interface.
- Displays:
  - Statistics summaries
  - Charts (bar, pie)
  - Formatted responses
- Handles user interaction only.

Controller Layer

- Receives HTTP requests.
- Routes queries to appropriate services.
- Uses middleware for authentication and validation.
- Does not process raw data directly.

Business Logic (Service Modules)

The core modular structure includes:

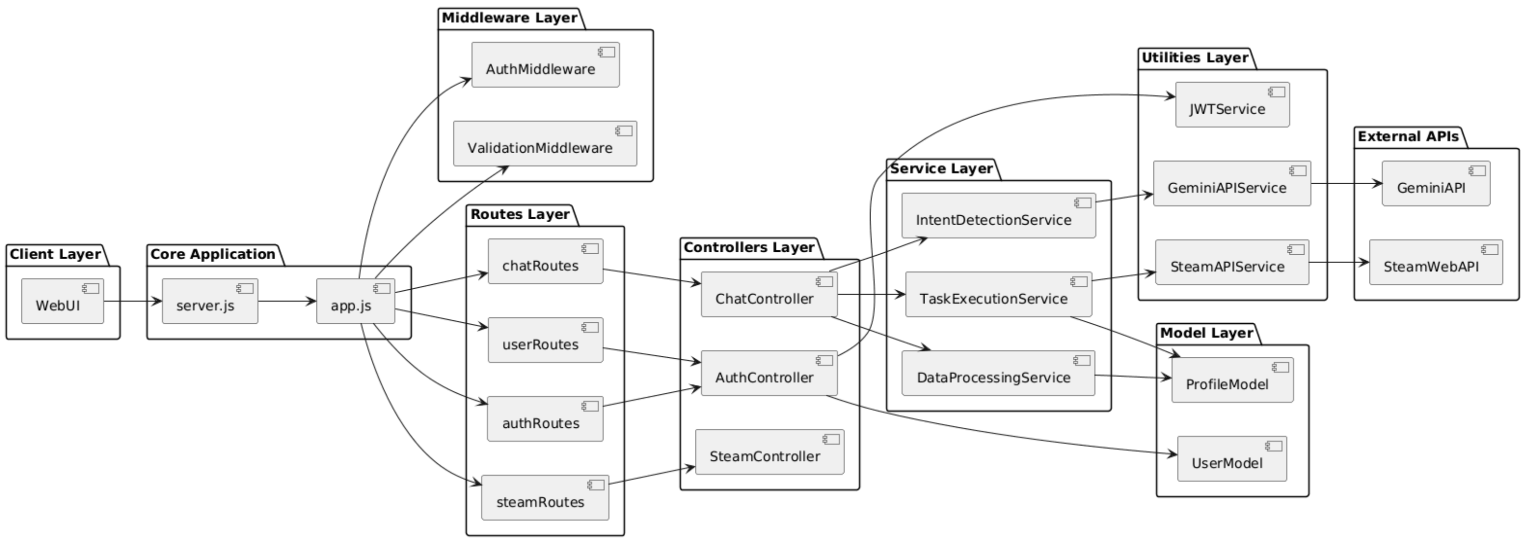
- Intent Detection Module (Gemini API integration)
- Validation Module (authentication & Steam ID checks)
- Task Execution Module (maps intent to logic)
- Steam API Integration Module
- Data Processing & Formatting Module

Granularity:

- **Coarse-grained modules:** Authentication, Intent Detection, Data Retrieval.
- **Fine-grained utilities:** Password hashing, API wrapper, chart generation.

Feature Expansion Mechanism:

- New functionality is added by:
    - Defining a new intent.
    - Adding a corresponding task handler.
- No modification to existing modules required.
- Ensures modularity and maintainability



.

B. Why MVC Architecture is the Best Choice (5 Marks)

The Steam Analytics Chatbot requires structured separation between UI, request handling, external API integration, and data processing. MVC supports this separation while enabling scalable intent-based routing.

- Maintainability
    - Clear separation of concerns.
    - Changes in UI or database do not affect business logic.

- Extensibility
    - New query types added via new intents and task handlers.
    - Existing architecture remains unchanged.

- Scalability
    - Modular services can expand independently.
    - External APIs are isolated from core logic.

- Performance
    - Node.js asynchronous execution.
    - Efficient handling of API calls and JSON processing.

- Security
    - Secure password hashing.
    - Middleware-based authentication.
    - Secure API key storage.

II. Application Components (10 Marks)

The following components are present in the system:

- Authentication Component

    - User registration and login
    - Password hashing
    - Session validation

- Query Handling Component

    - Receives user queries
    - Routes requests internally

- Intent Detection Component
    - Integrates Gemini API
    - Classifies user queries
- Validation Component
    - Checks authentication
    - Verifies Steam ID linkage
- Task Execution Component
    - Maps intent to specific tasks
    - Executes rule-based logic
- Steam API Integration Component
    - Retrieves gaming statistics
- Data Processing Component
    - Processes JSON data
    - Converts playtime values
    - Generates charts
- Database Component
    - User model
    - Profile model
- Middleware Component
- Authentication
- Validation
- Error handling

Final Summary

The Steam Analytics Chatbot follows MVC architecture enhanced with modular service design. The intent-based routing mechanism allows scalable and maintainable addition of new features without restructuring the core system, ensuring performance, security, and extensibility.
