Architecture Design Document
for
Teaching Assistant Management System

Group No. 7

Sai Ramana Reddy	CS17BTECH11022
Puneet Mangla	CS17BTECH11029
Vijay Tadikamalla	CS17BTECH11040
Tungadri Mandal	CS17BTECH11043

Contents

1 Overview

1.1 System Overview

Teaching Assistant Management System (TAMS) intends to facilitate TA allocation and management taking into account both students and professors' preferences. After successful TA allocation, the system can be used by professors as well as TAs to assign and submit course related tasks respectively. Upon course completion and obtaining feedback from professors, TA certificates and compensation are issues to the respective TAs.

1.2 System Context

The system context is defined clearly in the SRS. Basically, the TA allocation Database/Record (mentioned in DFD) is the main sink of the information as it will contain the final allocation and other necessary information like feedback. The main sources of information are the Professors (who provide information about their courses and TA preferences) and students (who provide information about their courses preferences).

1.3 Stakeholders of TAMS

The main stakeholders for the system are the individual users (i.e Students and Professors) who might use the system and the system designer/builder who will build and manage the TAMS. The main concerns of the two stakeholders are:

- **For Users:** The usability of the system. Response time should also be reasonable. Algorithm for TA allocation should be such that it can easily be changed with a better algorithm later.
- **For designer/builder:** The system should be able to handle load during course registration periods and is easy to modify, particularly to handle future extensions mentioned in the SRS (i.e. automated verification of student info, automated system for compensation etc.)

Hence, the key property for which the architecture is to be evaluated is the modifiability or extensibility of the system. Response time performance is another factor for which the system needs to be evaluated.

1.4 Scope of this Document

In this document, we describe two possible architectures for TAMS, compare them for various quality attributes, and then choose the most appropriate one, which is our final proposed architecture for TAMS. By discussing the two alternatives, we also provide the rationale for selecting the final architecture. For architecture, we consider only the component and connector view. To obtain two different architectures, we divided our team in two subgroups.

1.5 Definitions and Acronyms

Definitions:

- Google Sign-In: On some apps, a user can use Google account to sign in the app. In this process, Google APIs (OAuth 2.0 protocol) is used for authentication and authorization.

- Course Constraints: It is the exhaustive list of TA requirements and eligibility constraints specified by the professor for a particular course. For example, no. of TA required, CGPA bar, course prerequisites, student department and year.

Acronyms and Abbreviations

- TAMS: Teaching Assistant Management System
- TA: Teaching Assistant.
- GUI: Graphical User Interface.

2 Architecture Design and Analysis

2.1 Architecture 1 : Shared Data Architecture

Shared Data (Repository style) architecture consists of two component types – data repository and data accessor. Here the data repository acts as a passive entity or store. The data accessor of the repository are active and control the logic flow of the system. These data accessors check the central repository for changes and act according to the information as required.

In our architecture,

- Data repository contains student profile, student TA preferences, course information, task details which need persistent storage. It can also be thought as a combination of a Database and Server.
- Data accessors are separate modules built on top of this repository to manage student information, preferences, feedback, tasks, login, and feedback.

The diagram below shows this architecture:

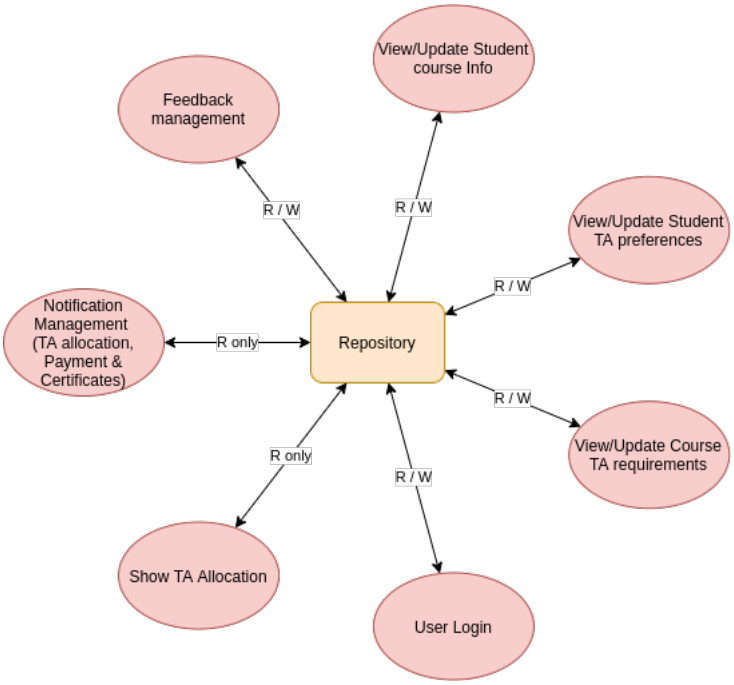

Figure 2.1: Shared Data Architecture

2.2 Architecture 2 : Client Server Architecture (3 Tier)

Client Server Architecture is distributed architecture where components are present on different platforms. These components cooperate with one another over a communication network to achieve a specific objective.

We use a 3-tier client–server architecture. It has the following components:

- **Client-tier:** It is the topmost layer of the architecture through which the users interact with our web application (GUI on browser). The primary function of this layer is to communicates with other tiers and translate the tasks and results to something that user can understand.
- **Middle or Application Tier:** It coordinates the application, processes the commands, makes logical decisions, evaluation, and performs calculations. It controls an application's functionality by performing detailed processing. It also moves and processes data between the two surrounding layers.
- **Database tier:** In this tier, information is stored and retrieved from the database or file system. The information is then passed back for processing and then back to the user.

By separating an application into tiers, we obtain the option of changing or adding a specific layer, instead of reworking the entire application. This helps in creating modular and reusable applications.

The diagram below shows this architecture:

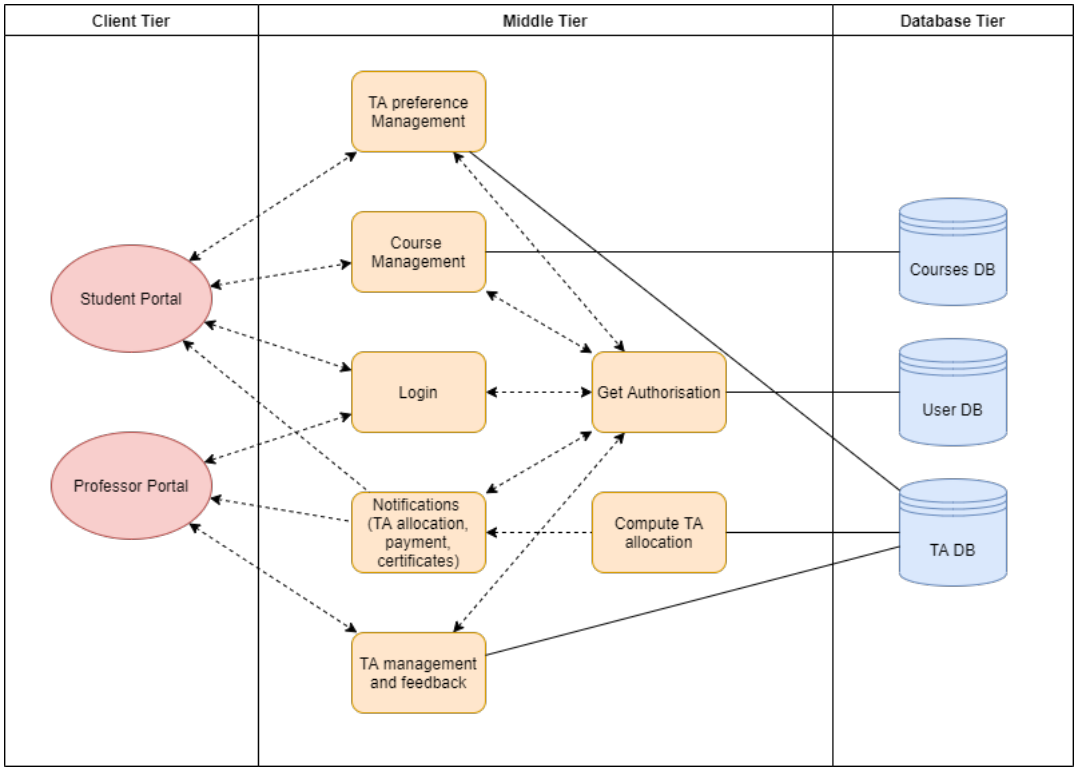

Figure 2.2: Client Server Architecture (3 Tier)

2.3 Comparing the Architectures

To compare between the above two architectures, we use the ATAM analysis method. The table below lists the various scenarios and the corresponding advantages and disadvantages for each architecture.

Criteria	**Shared Data Architecture**	**Client-Server Architecture**
Changes to Database layer	Difficult; Needs to modify all the accessors	Relatively easier; Requires modifying affected services in the Application tier
Data Security	Less secure; All modules are accessing the database	Secure, as only internal services can access the database
Changing TA allocation strategy	Easy	Relatively Easier
Work division among developers	Hard	Easy
Code Infrastructure setup	Easy	Difficult; The architecture is relatively more complex
Debugging	Difficult	Easier
Code duplication	High; All modules need to interact with database	Less duplication
Maintenance and support	Easy	Easier; Less code duplication and easier debugging helps to maintain code and provide long term supports
Adding additional features	Easy	Easy

3 Final Architecture

From the above comparison we can observe that our Client Server architecture is better than the Shared Data architecture on a number of points. It would be very difficult to make changes to the database layer as we would have to modify all the accessors in the Shared Data Architecture. It would also be less secure, reliable and would require more work for debugging. There is also significant code duplication involved which may slow down our application. Compared to this, our Client Server architecture is more secure, reliable and faster while having significantly less code duplication. Adding additional features would be as easy on both of the architectures.

So, after a comprehensive evaluation done using the ATAM analysis method, we conclude that the Client-Server Architecture (Architecture 2) is better.

4 Future extensions

In the current architecture, we send email notification regarding TA allocation status to all the user (Professors and students). So, as a future extension, we plan to include the Publish-subscribe architecture style in the existing architecture. This addition would allow the users to customize the notifications according to their preferences. This would also simplify the procedure of adding new notification systems like notification for TA vacancy, or availability of new course for TAship.