

Dummy Registration System.

Feb 2023

Kareen Ziadat

Contents

Part 1: Concept Statement.....	4
Main purpose	4
Functionality	4
Intended users.....	5
Technology / tools to be used.	6
Challenges that may be faced.	7
Special feature of the application.	8
Part 2: System Planning.....	9
Project Scope:.....	9
Description and purpose	9
Benefits	9
Alignment with corporate goals	9
Project goals:	10
Timeline:	11
Gantt chart.....	12
Cost estimation:.....	13
Development Costs.....	13
Technology and tools:.....	14
Testing, training, and support:.....	14
Total estimated costs before profit and contingency:.....	14
Profit and Contingency:	14
Risk management:	15
User adoption	15
Technical complexity.....	16
Data integrity	16
System performance.....	17
Security and privacy.....	17
Risk monitoring strategy.....	17
Cost risk.....	17
Part 3: Specification.....	18



Functional requirements:	18
Non-functional requirements:	18
User requirements:	19
System requirements:	19
Trello for requirement engineering.....	19
Part 4: System Design.....	20
Use case diagram:.....	20
Sequence diagrams:	21
Class Diagram:	23
State Diagram:	24
Part 5: Architectural design pattern.....	25
Architectural design pattern chosen:.....	25
Project's architecture, subsystem, and modules.	26
Project Architecture.....	26
Subsystems and Modules	27
Project's uniqueness in terms of design.	29
Language and framework used.....	30
Explanation of database	31
ERD.....	32
Part 6: Teamwork and implementation	33
Challenges faced while working in the team.	33
Description of how multidisciplinary team skills can produce an exceptional solution.	35
Part 7: Ethical and social responsibilities.....	36
Identification of ethical and social responsibilities	36
Main ethics principles:.....	36
How ethical and social responsibilities are important to software engineering and software development.	37
ACM/IEEE code of ethics	38
Reflection of ACM/IEEE on my solution	40
Referances	41

Part 1: Concept Statement.

Main purpose

The Dummy Registration System's main purpose is to offer a **user-friendly** platform for students to input their course preferences (sections) into a timetable. This is helpful for the university to collect thorough analytics on students' preferences prior to registration. These analytics will assess the most popular courses and establish the best time slots to accommodate as many students as possible (The approach to this is knowing that it is impossible to satisfy all students, and rather focusing on the majority). This data-driven strategy allows efficient allocation of resources (classrooms, and instructors) and guarantees that the courses offered are relevant to the student population's interests and demands. Furthermore, this analysis guarantees that the courses are correctly assigned to instructors. This not only benefits students but also assists the university to successfully manage the workload of instructors. Another benefit is its ability to eliminate frequent registration concerns such as schedule conflicts. By allowing students to pre-register for courses, it allows more time for the dean to discover and resolve scheduling problems prior to the actual registration. This proactive strategy reduces the stress of conflicting course scheduling.

Note: the days and times that each course might be offered are specified by the dean/head of department through their interface in the dummy registration system.

Functionality

The student should be able to log in using their username and password used for the portal account.

The registration system should have a user-friendly interface that is easy to navigate. It should provide a clear timetable layout where students can see a plus button in each time slot which when pressed, the students will get a checkbox list of all the courses that are still not completed for the student, prerequisite is met and available at that time. After the checkbox(es) is(/are) checked the name of the course(s) will display in the timetable.

	Sunday	Monday	Tuesday	Wednesday	Thursday	Saturday
8:30 am - 10:30 am	⊕	⊕	⊕	⊕	⊕	⊕
10:30 am - 12:30 pm	⊕	⊕	⊕	⊕	⊕	⊕
12:30 pm - 2:30 pm	⊕	⊕	⊕	⊕	⊕	⊕
2:30 pm - 4:30 pm	⊕	⊕	⊕	⊕	⊕	⊕
4:30 pm - 6:30 pm	⊕	⊕	⊕	⊕	⊕	⊕
6:30 pm - 8:30 pm	⊕	⊕	⊕	⊕	⊕	⊕

Figure 1: Empty timetable template

After the student has completed the dummy registration, they should be able to save their choices by clicking the save button under the timetable.

To add, the student can specify which course is the top priority and which is high priority in a simple drop-down list.





 Top Priority	
 High Priority	

Figure 2: Priority matrix dropdown

Top priority is equivalent to 5 points and the high priority is equivalent to 3 points.

From the dean's/department's side: The system should display the results of the analysis and allow them to specify the days and times that each course might be offered in.

For the analysis, the first table will have the course id, name, average priority (numerical value), interest count (number of the students interested in registering the course at any time), course urgency (how important it is to offer this course in the upcoming year) which is calculated using a special formula: $\text{Course Importance} = (\text{Average Priority} \times \text{Priority Weight}) + (\text{Interest Count} \times \text{Interest Weight})$ where priority weight is 0.7 and interest weight is 0.3.

The second table has the course id, course name, option day, option time and interest count sorted by interest count. This table can be accessed alone or by pressing any course id (hyperlink in first table) and they will get the second table filtered by the course chosen.

Both tables can be filtered by school, department, course id.

They can also view, add, modify, and remove section options.

Intended users.

Students who will log in using their existing portal account credentials, can select and manage the course for each day and time, and will specify which courses are off higher priority.

University administrators: who will enter the data about the days and times in which the courses might be offered, and they are given useful insights to help them manage the registration process effectively. The system displays the useful tables mentioned above, allowing for more efficient resource allocation and scheduling considerations.

Instructors aren't direct users but as mentioned before they will specify which courses they can teach and specify the constraints of when they can offer the course (date and time). To add, they will benefit from the system by successfully managing their workload.



Technology / tools to be used.

1. **Trello** will be used to stay on top of the project, ensure tasks will be allocated to the right people, improve communication, and ensure time management.
2. **Github** is required for effective collaboration, version control, and quality assurance. Github offers version control capabilities which lets many developers work on the project concurrently without conflict and makes code review and improvement easier. It also includes issue tracking and documentation tools which will help structure and streamline the development process.
3. The system will be **developed** as a web application using HTML, CSS, and JavaScript for the user interface. These technologies will provide a user-friendly interface and responsive design for access on various devices.
4. **JavaScript** will handle user interactions, dynamic content changes (ex. updating the timetable and displaying the course list) and sending **AJAX** requests to the server without requiring a page refresh. Note: Ajax allows web applications to send and retrieve data from a server asynchronously without interfering with the existing page's appearance and behavior.
5. **Visual Studio Code IDE** will be used because it supports all required languages and frameworks, adheres to the MIT license (no restrictions on software usage and distribution, making it ideal for educational purposes), and has Git integrated, allowing for seamless integration with GitHub (developers can perform Git commands ex. commit, pull, and push changes to the repository, streamlining the development process). Furthermore, it includes an easy-to-use interface, a large library of extensions, AI integration, debugging support, and so on.
6. **PHP**: will handle server requests, process user interface data (ex. Increment interest count value when checkbox is selected), interact with the MySQL database (data storage, and retrieval), and return responses to the client. It also handles sorting and filtering algorithms and analyzes the results of DRS by calculating priorities and creating analytics for the dean's report.
7. **MySQL database management system (DBMS)** will be used to create internal tables that will store, retrieve, and manage data related to course options, priorities, and store results of analysis.
8. **LDAP** will be implemented to enable student login using their existing portal account credentials (existing active directory) to securely authenticate and authorize students. It is an open source, lightweight and secure protocol making it suitable.
9. The system will be deployed on **Amazon Web Services (AWS)** cloud platform which will ensure scalability, reliability, and maintenance.



Challenges that may be faced.

1. It may take time and effort to familiarize users with the system and how to use it, especially since it is a new concept.
2. The system heavily relies on accurate course information like which courses will be available and at which time which might not be known and maintaining data integrity pose challenges.
3. The system needs to interact with other university systems, such as the student portal and providing seamless integration and data synchronization across these systems might be hard.
4. It may be essential to provide videos or training sessions to ensure successful adoption and usage.
5. It is hard to balance the complexity of the system's functions with a user-friendly interface. The system should be simple enough for all users, regardless of technical ability (simple interface) but still capable of managing complex operations such as course selection, priority setting, and data analysis. (abstraction)
6. Scalability might be hard to achieve. As the university students increase, study plans vary, and data increase, the system must scale to accommodate, ensuring that the system runs quickly during peak times (before formal registration) and minimizes downtime.
7. Creating the system can be time-consuming and expensive so balancing the budget and resources is difficult, especially in a university context when funding is limited.
8. Relying on specific technologies and platforms (such as AWS, LDAP, or the programming languages used) can cause problems if they become unsupported or have compatibility concerns with new updates.
9. Since the system is transitioning from traditional approaches to a new, technology-driven approach, this requires management in the cultural change. (overcoming resistance to change)



Special feature of the application.

DRS collects information from student course selections in a user-friendly interactive way rather than the traditional way of selecting from a table which can be tiring and confusing going back and forth between all the records. It determines the most popular courses and the optimum time slots and takes into consideration the importance of registering the course in the next semester which makes sure that even if a course has a low of students interested in however, they all need it to graduate, the dean will be aware of this prior to actual registration. This ensures that resources are allocated efficiently (classrooms, instructors) and that course offerings are aligned with student interests and expectations.

To add, the dean or head of department plays an important role in fine-tuning the course calendar. They will set days and time slots for each course, increasing system efficiency and reducing potential disappointments. This method is especially effective when an instructor is only available at certain hours and/or times. Limiting the selections available to students decreases the possibility of disappointment and guarantees that the DRS provides the most relevant and possible course options.



Part 2: System Planning.

Project Scope:

Description and purpose

The Dummy Registration System (DRS) is a web-based system designed to revamp the course registration process for students and administrators at the university. Its primary goal is to provide a user-centric platform for students to pre-register and manage their course selections in a dynamic and interactive approach. This helps in the collection of analytics on student preferences, as well as in the efficient scheduling and distribution of resources. The purpose of DRS is to improve student satisfaction and operational efficiency at the university.

Benefits

Students: An easy and flexible way to plan their upcoming semester schedule.

Administrators: The dean and head of department will get insightful data that will reduce the headache and complexity of course planning, improve resource allocation (classrooms, labs, instructors) since the demand of courses is clear.

Instructors: Better workload management. Example, if a course has a high interest and not many instructors teach it, it is wise to recruit more instructors to ensure that instructors are not overworked ensuring job satisfaction, productivity, and teaching quality.

Alignment with corporate goals

The Dummy Registration System (DRS) aligns with university goals of improving the student experience by providing a simple and efficient pre-registration procedure. Moreover, it embodies technological innovation by encouraging effective resource use and data-driven decision-making, both of which are critical for modern educational institutions. To add, DRS promotes effective academic planning assisting in the attainment of every university's strategic goal like student retention and graduation rates.

While this is not included in the project scope, including AI in the future to provide course recommendations for students and sections recommendation for head/dean will improve the effectiveness of the system.



Project goals:

1. **Strategic decision-making:** Comprehensive analytics on student course preferences will be provided by the system which will help administrators make evidence-based decisions on course scheduling.
2. **Optimizing resource allocation:** The system improves course coordination which will result in a more efficient use of university resources, such as classrooms, faculty time, and administrative efforts, guaranteeing that the most wanted courses are delivered at the most convenient times, maximizing resource utilization, and avoiding wasteful spending.
3. **Reducing registration-related stress:** The solution will minimize stress of not finding the appropriate courses or times to register to by offering the most relevant courses at the most asked time.
4. **Improving workload management:** By offering insights into course demands and student preferences, it will enable faculty to better distribute courses resulting in a more balanced workload and higher job satisfaction.
5. **Maximizing student academic success:** By allowing students to easily plan and visualize their academic schedules, the system assists students in making educated decisions that match their academic and career goals. This possibly results in better academic outcomes.



Timeline:

1. **Project initiation and planning:** Jan 1, 2024- Jan 15, 2024
Defining project scope, objectives, and forming the Agile team.
2. **Requirement gathering and analysis:** Jan 16, 2024- Feb 15, 2024
Gathering detailed functional and non-functional requirements from stakeholders, including students (through a quick survey), faculty (through a separate survey), and administrators (interview with dean, heads of department).
3. **Design phase 1: System architecture & UI** Feb 16, 2024- Mar 15, 2024
Initial design of system architecture, login interface based on functional requirements gathered in previous phase.
4. **Development sprint 1: Core functionalities & security** Mar 16, 2024- Apr 15, 2024
Development of core functionalities including user authentication (LDAP setup), basic UI for dean/head of department, and develop system databases using mySQL. (Release 1)
5. **Testing and feedback 1: Initial Functionalities & Security** Apr 16, 2024- Apr 30, 2024
Testing of initial functionalities, security measures and gathering feedback.
6. **Design and development sprint 2:** May 1, 2024- Jun 1, 2024
Enhancements based on feedback, developing student's interface design and functionalities (preregistration, and priority feature), focusing on scalability and performance optimization. (Release 2)
7. **Testing and feedback 2: New features & performance** Jun 2, 2024- Jun 15, 2024
Testing of new features and second round of feedback.
8. **Design and development sprint 3:** Jun 16, 2024- Jul 15, 2024
Developing dean/head of department interface design and functionalities (analytics dashboard, course options, searching and filtering). (Release 3).
9. **Final testing and feedback 3:** Jul 16, 2024- Jul 31, 2024
Complete usability testing with a student pool, database testing with real data from university (database provided by university), bug fixes, and final feedback incorporation.
10. **Deployment preparations: Strategy & User Training** Aug 1, 2024- Aug 15, 2024
Finalizing deployment strategy, user training materials, user, and system support documentation, disaster recovery plans.



11. **Deployment: System Go-Live** Aug 16, 2024- Aug 31, 2024
University will receive the system, implementation of training and support.
12. **Post-deployment support:** Sep 1, 2024- Sep 30, 2024
Monitoring system performance, gathering feedback, and making necessary adjustments.
13. **Project Closure:** Oct 1, 2024- Oct 15, 2024
Formal project closure, and reflective analysis, final budget review.
14. **Ongoing maintenance and updates:** Oct 16, 2024-
onwards Regular system updates, maintenance, and continuous improvement based on user feedback. Conduct Biweekly security editing and monthly vulnerability scans.

Gantt chart

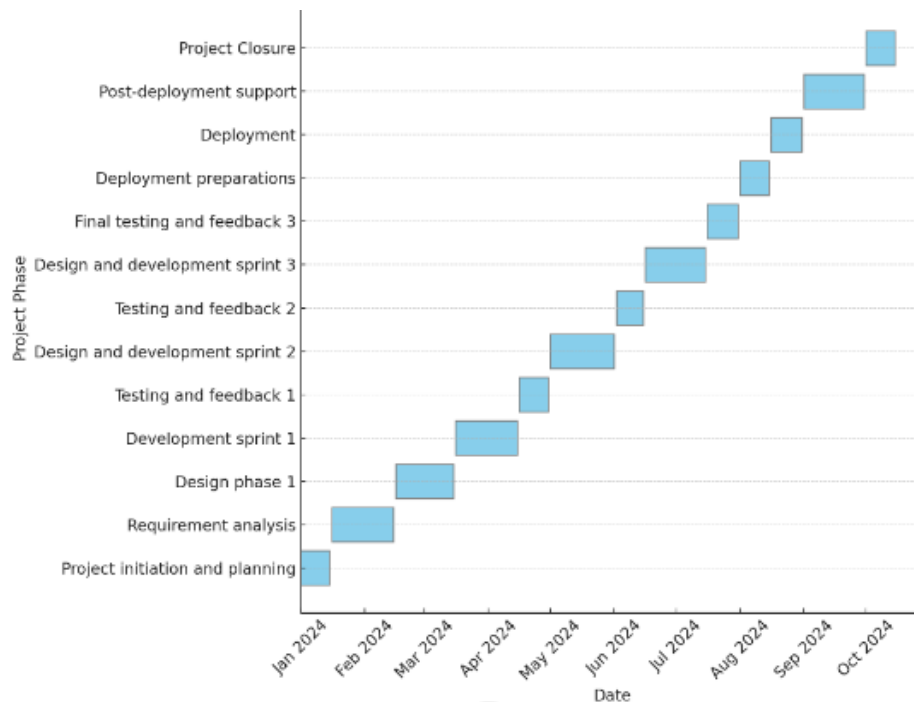


Figure 3: Gantt chart showing timeline.



Cost estimation:

Development Costs

Personnel: The team of developers would be 5 to ensure effective communication and to fully benefit from the agile principles. To ensure cost isn't too high (budget for university system), students will be involved to work as interns who can work at a reduced cost or as part of their academic.

The development team is composed of 1 senior developer, 2 mid-level developers, and 3 Junior developers (interns from the university). Each with a different role:

1. **Senior Developer:** Oversees project, coordinates development, and ensures quality and adherence to objectives.
2. **Mid-level Developers:** Contribute significantly to both front-end and back-end development.
3. **Junior Developers/Interns:** Help with development, mostly with easier tasks, testing, and minimal documentation.

Salary Estimation for development team:

Senior Developer: Part-time work, around **2,000** JDs for the project duration.

Mid-level Developers: They will work by contract which specifies the features (according to functional requirements) and time of delivery (according to the timeline). Estimated Salary per developer is around 1,200 JDs each for the project duration. So, a total of **2,400** JDs for 2 Developers.

Junior Developers/Interns: Each intern's estimated salary is 500 JDs, considering that they will work in return for academic credit (internship). For the three interns the total would be **1,500** JDs.

There is an additional cost for initial training sessions, particularly for interns which might be around **200** JDs.

The team is structured in a way to achieve a balance between expertise and cost-effectiveness. The senior developer assures quality and project leadership, mid-level developers give a solid development foundation, and interns contribute to development and get vital expertise while lowering labor costs. Moreover, not all developers are working on this project full-time, which results in good budget management. Moreover, using interns in a university setting is a strategic decision to save money while giving students real-world experience. The budget for the development team is reasonable considering the project's duration and scope of work.

Estimated cost: 6,100 JDs.



Technology and tools:

Project management tool (Trello, GitHub): 0 JDs cost (open source).

Frontend web development HTML, CSS, JavaScript, AJAX: 0 JDs cost (open source).

PHP for backend: 0 JDs cost (open source).

Database management system (MySQL): 0 JDs cost (open source).

LDAP for secure login: Minimal to moderate costs.

AWS: 500 JDs (assuming minimal to moderate costs for servers and cloud services).

Estimated Cost: 500 JDs.

Testing, training, and support:

System testing:

- Usability testing: TryMyUI which offers the chance to test the site either with anonymous reviewers, or our own pool of candidates. The list of features includes video screencaps, mouse clicks and keystrokes, voiceover commentary and written feedback, system usability scale questionnaire, mobile testing \$99 per month for 2 weeks from July 16 till July 31, 2024 overall: \$99 = **70 JDs**.

Training materials: For users and administrators. 120 JDs.

Post-deployment support: For monitoring and adjustments. 100 JDs.

Estimated Cost: 290 JDs.

Total estimated costs before profit and contingency:

Development Costs: 6,100 JDs

Technology/tools: 500 JDs

Testing/support costs: 290 JDs

Subtotal: 6,890 JDs

Profit and Contingency:

Contingency: any hidden or unforeseen costs that might arise during the project's lifecycle.

Profit (10% of Subtotal): 689 JDS

Contingency (15% of Subtotal): 1033.5 JDs

Overall, Subtotal is 6,890 JDs, Profit is 689 JDs, Contingency is 1,033.5 JDs so total cost is 8,612.5 JDs.



Risk management:

This plan is intended to anticipate and address possible issues before they have a negative influence on the project. It should be reviewed on a regular basis to update risk assessments, strategies, and plans regarding the most recent project developments and feedback.

5x5 Risk Matrix Example

		Impact <i>How severe would the outcomes be if the risk occurred?</i>				
		Insignificant 1	Minor 2	Significant 3	Major 4	Severe 5
Probability <i>What is the probability the risk will happen?</i>	5 Almost Certain	Medium 5	High 10	Very high 15	Extreme 20	Extreme 25
	4 Likely	Medium 4	Medium 8	High 12	Very high 16	Extreme 20
	3 Moderate	Low 3	Medium 6	Medium 9	High 12	Very high 15
	2 Unlikely	Very low 2	Low 4	Medium 6	Medium 8	High 10
	1 Rare	Very low 1	Very low 2	Low 3	Medium 4	Medium 5

Figure 4: Risk management matrix used.

User adoption

Risk assessment:

Probability: Moderate (3), the system may be hard to use for some people because of too many course and section options.

Impact: Significant (3) since poor adoption will make the system ineffective.

Score: 9 (Medium 9)

Mitigation strategies:

Develop an outreach program that includes instructional videos, and Q&A sessions.

To get early input, launch a pilot program with a group of students.

Contingency plans:

Establish a helpdesk with extended hours during the initial phase.

Prepare digital user guides for quick reference.



Technical complexity

Risk assessment:

Probability: Likely (4) since we are developing custom features which can cause unforeseen issues.

Impact: Major (4), delays or failures will affect the entire software development process.

Score: 16 (Very high 16)

Mitigation strategies:

To discover problems early, use an iterative development approach (agile) with continuous deployment.

Consider conducting pair programming sessions to ensure quality.

Include buffer times in the project timeline to account for unexpected technical issues.

Contingency plans:

Maintain a rollback strategy in case of critical issues following a change to go back to the previous stable version. (use of GitHub for version control)

Data integrity

Risk assessment:

Probability: Moderate (3), the system relies on external databases.

Impact: Major (4), Incorrect data results in poor decision-making and ineffective system analytics.

Score: 12 (High 12).

Mitigation strategies:

Follow strong data governance policies.

Conduct regular checks.

Contingency plans:

To maintain data consistency, use manual supervision throughout the initial phase.



System performance

Risk assessment:

Probability: Unlikely (2) since cloud services are usually robust.

Impact: Major (4) system credibility will suffer if it can't work well in peak registration times.

Score: 8 (Medium 8)

Mitigation strategies:

Perform load testing to simulate peak usage and identify issues.

Use a cloud service provider with auto-scaling features. (AWS is used)

Contingency plans:

Implement a load balancer to distribute traffic evenly across servers.

Security and privacy

Risk assessment:

Probability: Moderate (3) since only information on courses pre-registered are available.

Impact: Minor (2) breaches don't have a significant effect since data stored can't be used maliciously.

Score: 6 (Medium 6)

Mitigation strategies:

Encrypt data and conduct penetration testing and security audits periodically.

Contingency plans:

Create an incident response group with expertise in managing security breaches.

Risk monitoring strategy

1. Maintain thorough logs of all risk management operations to guarantee accountability.
2. Inform all parties involved on the state of the risk and include them in discussions regarding mitigation techniques.

Cost risk

Assigned a percentage of the total budget (15%) to a contingency fund to cover unexpected costs.



Part 3: Specification.

Functional requirements:

1. The students, deans, and heads of departments should be able to log in using their portal credentials and logout.
2. The student should be able to choose courses from a list of their remaining courses and add them to their timetable via an interactive timetable interface that displays days as column headings and university time slots as row headings. To choose the course, each cell in this timetable will have a plus button that once pressed will display a checkbox list of the remaining courses available in the day-time slot.
3. The student should be able to specify the course that is a top priority to them and the course that is of high priority to them.
4. The dean/head of department should be able to view the results of the preregistration in simple tables.
5. The dean/head of department can filter their tables by the school, course, and department and can sort them by course urgency, average priority, and interest count.
6. The dean/head of department can add, modify, and delete course options through a simple table and forms.

Non-functional requirements:

1. User-friendly and intuitive interface.
2. Capable of handling high traffic during peak registration times, with efficient load balancing and auto-scaling.
3. Able to accommodate a growing number of students and courses.
4. Responsive design for access across various devices and browsers.
5. Regular checks to ensure accuracy and consistency of course information.
6. All data in transit and at rest must be encrypted using industry-standard encryption protocols (AES-256 for data at rest, TLS 1.2 for data in transit).
7. The system should do backups overnight, save in two different physical locations using AWS Backup.
8. The system should be responsive on all devices.



User requirements:

1. Students:
 1. Should not find it difficult to navigate and understand the website.
 2. Should be able to log in using existing portal credentials.
 3. Should be able to log out.
 4. Should be able to recover forgotten password.
 5. Should be able to manage the timetable.
 6. Should be able to set a top priority to a course.
 7. Should be able to set a high priority to a course.
2. Administrators:
 1. Should be able to log in using existing portal credentials.
 2. Should be able to log out.
 3. Should be able to view analytics on course preferences and registrations.
 4. Should be able to add a course option.
 5. Should be able to view all course options.
 6. Should be able to modify course options.
 7. Should be able to remove course options.
3. Instructors:
 - Not direct users, but they provide input on course availability and constraints.

System requirements:

1. System should be user-friendly and intuitive.
2. System should ensure secure login.
3. System should make sure that it provides valuable analytics based on correct preregistration data.
4. System should be able to accommodate the growing number of students, sections, and courses.
5. System should be able to handle concurrent users, especially during peak registration times.
6. System should reflect the courses based on study plan and prerequisite correctly for each student.
7. System should ensure the courses are only preregistered in the time constraints specified by the system administrator.
8. System should backup data every night automatically.
9. System should adhere to GDPR.

Trello for requirement engineering.

IMPORTANT: a detailed description of all the functionalities is found in the Trello workspace created for this system <https://trello.com/b/MVb1auGb> .



Part 4: System Design.

Use case diagram:

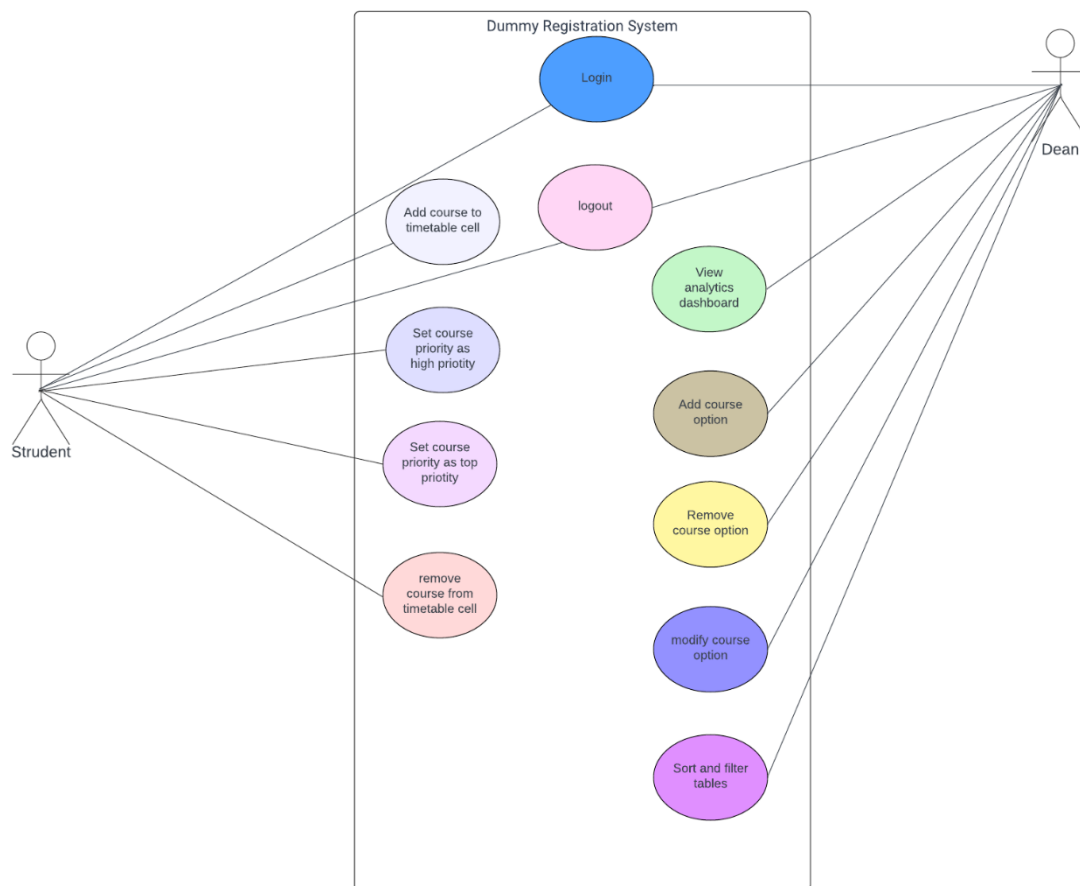


Figure 5: Use case diagram



Sequence diagrams:

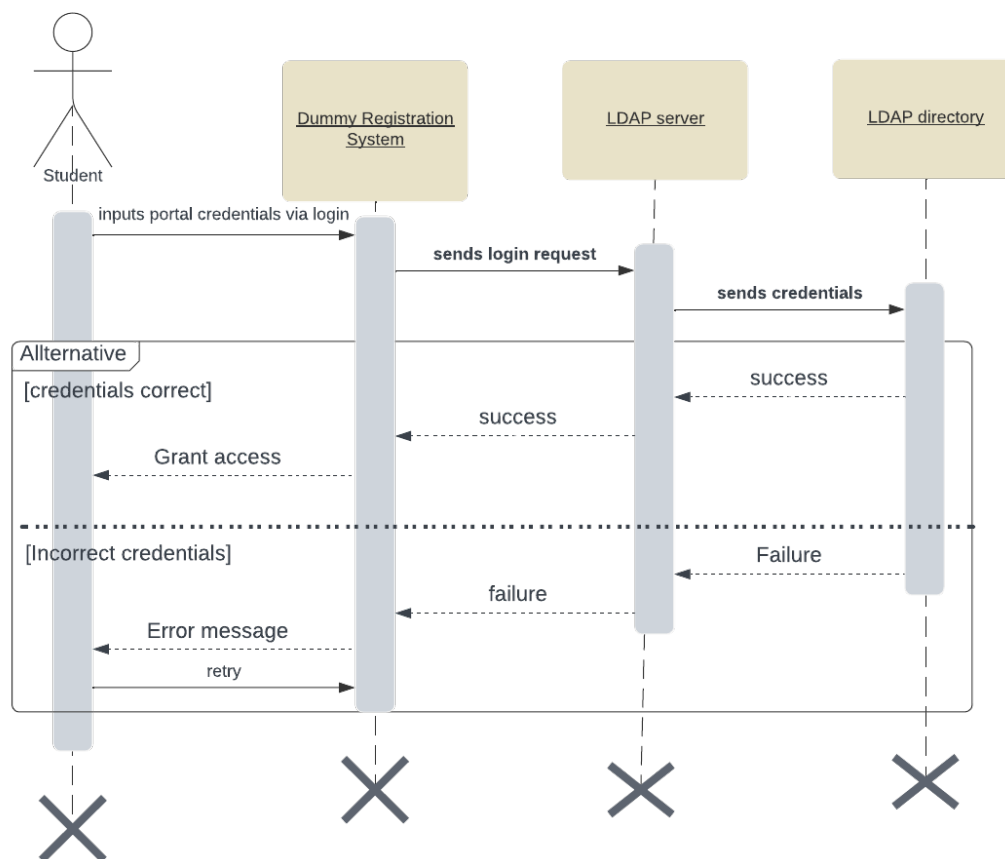


Figure 6: sequence diagram for login process of student/dean/head of departments.

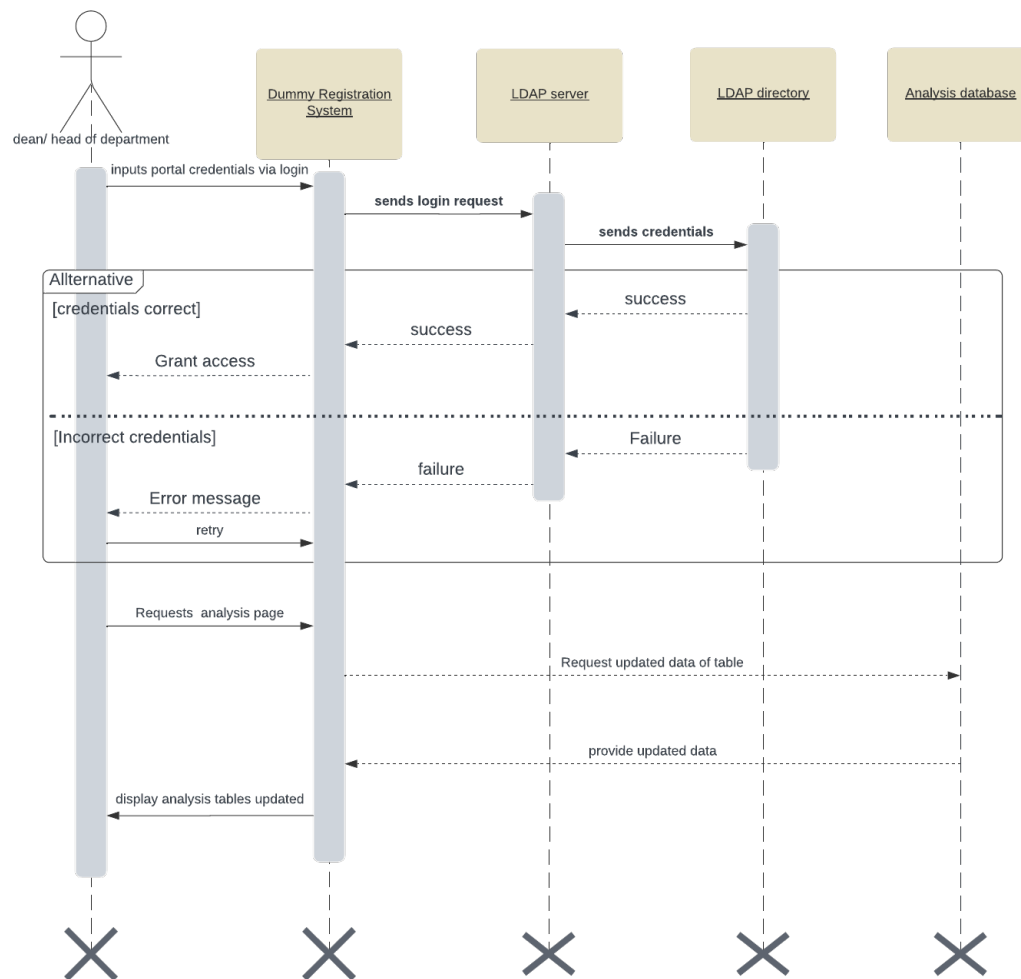


Figure 7: Sequence diagram showing dean/head of department getting analysis of DRS.



Class Diagram:

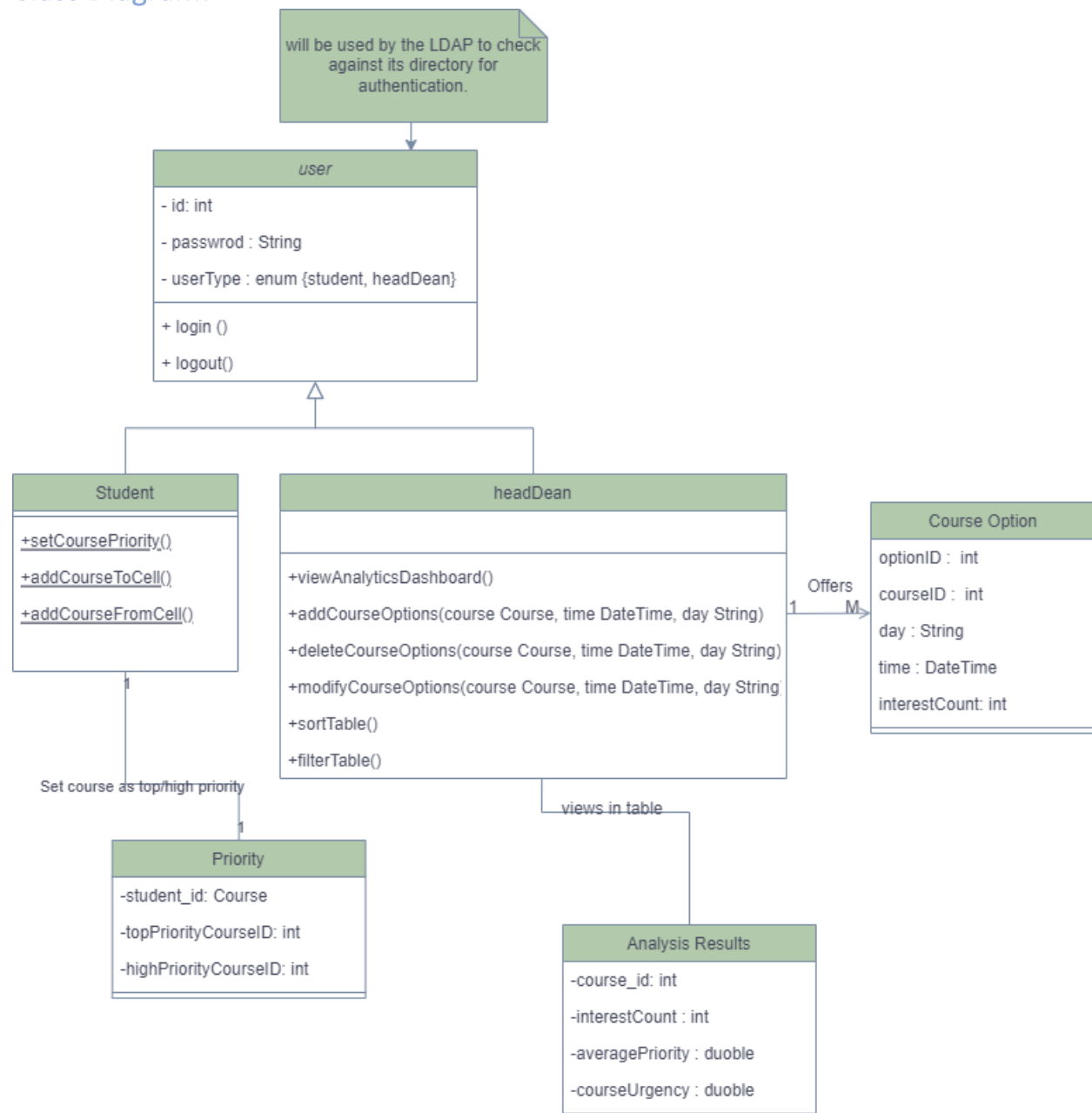


Figure 8: Class diagram for the system



State Diagram:

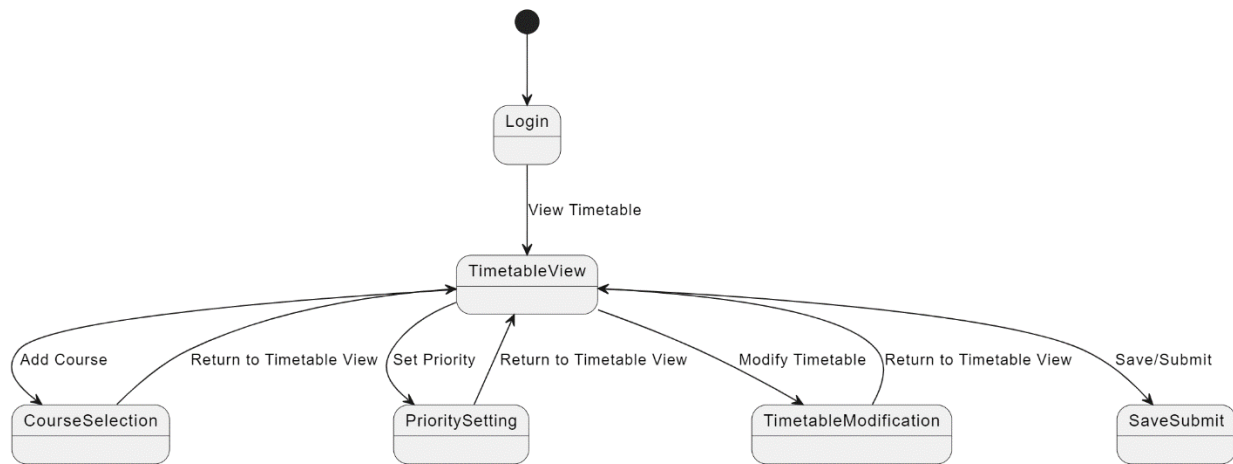


Figure 9: State diagram for students



Part 5: Architectural design pattern.

Architectural design pattern chosen:

Based on the application requirements, the most suitable architectural design pattern would be the Model-View-Controller (MVC) pattern.

Here's the justification for choosing the MVC pattern:

1. **User interface:** The system requires an interactive user interface that enables the timetable interface and real-time depiction of it. Because it separates the user interface (View) from the business logic (Model), the MVC pattern is appropriate. The UI can be interactive and easily updated without affecting the underlying data processing.
2. **Multiple user types:** The system is designed to be used by students and university administrators (deans and head of departments). The MVC design can provide customized views for each user type while keeping the controller logic and model structure consistent.
3. **Data-driven analytics:** The analytics dashboard for administrators is an essential part of the system. MVC's Model can be set up to handle analytics, while the View displays this data in a user-friendly way.
4. **Scalability and performance:** Given that the system must be scalable and should perform well during peak time. A key benefit of MVC design is that changes in one component have minimal impact on the others which allows for horizontal scaling (adding more resources) to be done more efficiently.
5. **Security and data integrity:** The Controller works as a gatekeeper between the View and the Model, it is easier to impose security features such as authentication and data validation when using the MVC model.
6. **Integration:** Because the Controller may be created to interface with these external systems without altering the Model or the View, MVC enables simpler integration with the LDAP server.
7. HTML, CSS, JavaScript, PHP, MySQL are all well-aligned with the MVC approach which is frequently used in web application development.
8. MVC lets components be adjusted independently so future requirements can be integrated easily.



Project's architecture, subsystem, and modules.

This is the project architecture, which is built around the MVC architectural pattern. Here's an overview of how the architecture might be organized:

Project Architecture

1. Presentation (View)

- **User Interface (UI):** HTML, CSS, and JavaScript are used to create web pages.
- **Student Interface:** For course preregistration and course prioritization.
- **Administrator Interface:** To view analytics dashboard.
- **Common Components:** Shared elements like headers, footers, and navigation bars.

2. Application (Controller)

- **User session management:** Handling user logins, sessions, and security.
- **Request handler:** Interprets user actions like pressing the add button, ticking checkbox, drop down list and communicates with the Model.
- **Input validator:** Before sending user input to the Model, it ensures that it matches the system's criteria.

3. Business Logic (Model)

- **User management:** Integrates with LDAP for authentication.
- **Course option management:** Manages (add, views, modifies, and deletes) course section options (course id, time, and day offered).
- **Preregistration:** Logic for displaying the correct courses in the checklist, adding and removing courses from timetable.
- **Analytics engine:** Processes, analyzes, and stores data in database to display on the dashboard.
- **Integration handlers:** Communicates with external databases, like the student's incomplete courses and the courses data.
- **Databases:** Course options (Contains details about courses, time, day), preregistration (saves results of preregistration for each student), Priority list (contains information on the student id, their top priority course and their high priority course), Analytics Data (course id, interest count, average priority, course urgency).



Subsystems and Modules

Subsystems:

1. Authentication:

Modules:

- Login: Verifies user credentials using LDAP server.
- Session: Manages user session and cookies.

2. Course registration:

Modules:

- Course prioritizing: Lets students prioritize courses.
- Schedule builder: Allows students to add courses into a timetable.
- Validation: Checks which courses from the incomplete courses for the use are offered in the day and time intersection selected (a valid option).
- Visualization: responsible for displaying the timetable layout, the courses checked in each cell.

3. Course sections:

Modules:

- Managing course section data: add, views, modifies, and deletes course section options (course id, time, and day offered)

4. Analytics:

Modules:

- Data collection: Gathers data from the student preregistration activities.
- Analysis: Performs mathematical calculations on data collected like count of people interested in the course, count of students interested in the course option, average of priority of the course and the urgency of offering the course in the next semester.
- Reporting: Generates analysis tables, allows sorting and filtering.

5. Integration:

Modules

- LDAP integration: Interacts with the LDAP server for authentication and authorization.

6. Data management:

Modules:

- Backup and restore: Manages data backup procedures and restoration in case of data loss.

7. User interface:

Modules:

- Front-end: A collection of common UI components and templates.

8. Infrastructure:

Modules

- Server management: is responsible for managing the application server.
- Cloud services: Manages interactions with Amazon Web Services (AWS) for hosting and scalability.
- Security: Ensures encryption and data security.

Each of these subsystems and modules will collaborate using the MVC pattern to make a cohesive system that meets the Dummy Registration System's functional and non-functional requirements. Individual components can be designed, tested, and maintained independently using the modular method, enabling flexibility and scalability.

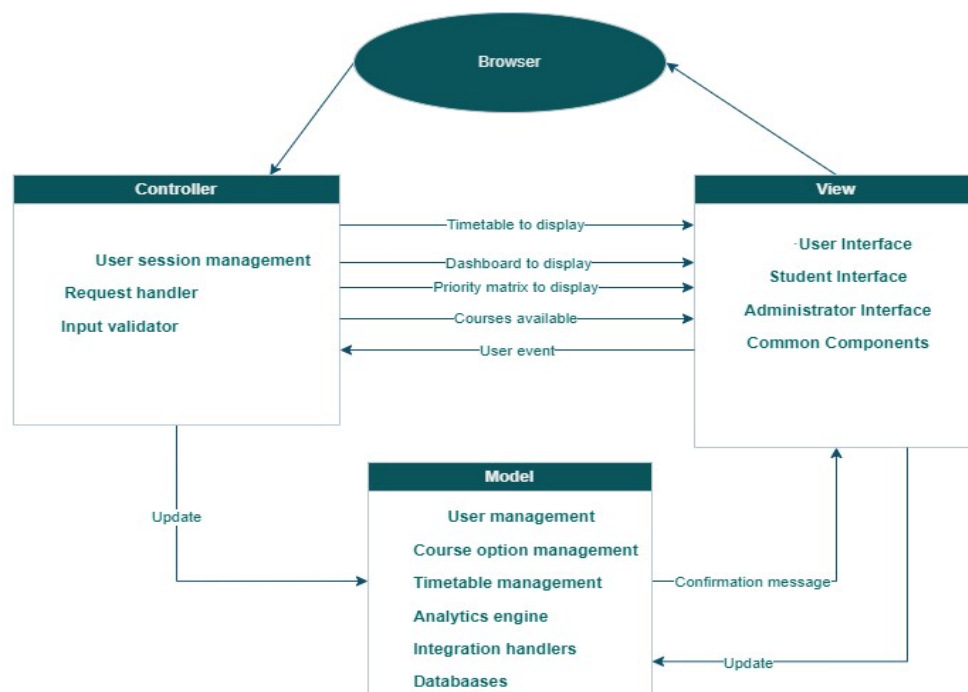


Figure 10: DRS architecture



Project's uniqueness in terms of design.

1. **Separation of Concerns:** SOC entails breaking down the software into different sections with unique roles and interfaces that can be built, tested, and altered independently. MVC pattern breaks the system down into three interconnected components: the Model (data and business logic), the View (user interface), and the Controller (intermediary between the Model and the View). SOC aids in the design and maintenance of modular, coherent, and loosely connected systems.
2. **Enhanced UI:** The MVC pattern enables a more dynamic and responsive user interface, which is critical for the timetable dynamic management capabilities and real-time visualization. This is possible since changes in the View can happen independently of the Model.
3. **Scalability and flexibility:** DRS is built to support a huge number of users and data and since MVC is modular, parts of the system (e.g. structure subsystem, data management subsystem, analytics subsystem, etc.) can be scaled separately as the number of students increase or requirements change.
4. MVC allows the creation of customized interfaces (Views) for each user type while keeping the underlying logic (Model and Controller) consistent. **Customized UX:** Different users (students, administrators) interact with the system in various ways.
5. MVC's Model is excellent for managing complicated data operations and analytics. It successfully manages the processing of student preferences and course data in DRS, making it easy to deliver this data to administrators via View.
6. **Easily maintained and updated:** When the system's various components are clearly separated, upgrading, or maintaining one part (such as changing the user interface or updating the data processing logic) does not demand an extensive rewrite of the system. DRS becomes more maintainable and flexible to future requirements as a result.
7. **Security and data Integrity:** The Controller in MVC serves as a gatekeeper for data flowing to and from the Model and View. Because the DRS handles sensitive student and course information, this structure is suitable for adding security mechanisms like authentication and data validation.
8. **Integration:** MVC simplifies connection with external systems (such as LDAP for authentication) since the Controller can manage these interactions without interfering with the Model or View.



Language and framework used.

1. Front-End:

- HTML: a markup language that is used to create and design web content.
- CSS: A style sheet language used to specify the design of the webpage. It is used to improve the look of the HTML structure, resulting in a more engaging and responsive user experience. (Framework recommendation: Bootstrap or Material-UI for a more consistent with less effort and it offers pre-designed components and a grid system that makes it easier to create a responsive design that adapts to different screen sizes, which is essential for students accessing the system on various devices.)
- JavaScript: allows incorporating interaction into website. It is used to handle user interactions, dynamic content updates (such as updating the timetable and presenting course listings), and AJAX (Asynchronous JavaScript and XML) queries to the server. (Framework recommendation: React.js which provides a more organized approach to developing interactive and dynamic user interfaces)

2. Back-End:

- PHP: A server-side scripting language used to handle server requests, analyze user interface data, connect with the MySQL database for data storage and retrieval, and execute sorting and filtering algorithms. PHP is also in charge of assessing DRS data and creating analytics for the dean's report. (Framework recommendation: Laravel, a modern and feature-rich PHP framework that could simplify many back-end development processes, like routing and sessions.)
- LDAP (Lightweight Directory Access Protocol): Used to provide safe user authentication to ensure safe access to the system. It communicates with the university's existing active directory to authenticate and authorize students and staff.

3. Database:

- MySQL: it is a commonly used DBMS for storing, retrieving, and managing data relating to course selections, priorities, and analysis findings. MySQL is dependable, and scalable making it an excellent solution for the DRS's database requirements. The data of DRS is a structured, needs complicated queries, and MySQL offers dependable transactional support, sophisticated tools for complex queries, and a solid foundation for interfacing with LDAP.

4. Cloud:

- Amazon Web Services (AWS): A cloud platform was chosen for system deployment. AWS supports the system's scalability, reliability, and maintenance, assuring its seamless operation and availability.



Explanation of database

1. External provided by the university: it is important to understand the structure of the university database to know how they can serve the website.
 - **LDAP directory:** will include information about the user's portal id, password, and role.
 - **Remaining courses database:** It lists courses each student still needs to complete. This table includes two attributes: Student_ID and Remaining_Course_ID. It's generated by the university as a special view. This view automatically calculates the courses a student hasn't yet completed, considering both their study plan and their current or completed coursework. It derives this information from several related tables: Students, Courses, Enrollment, and Study Plan.

A course appears in the Remaining Courses Table if it meets two criteria for a student:

 - The students have not yet completed the course, nor are they currently enrolled in it.
 - The student has already completed or is currently enrolled in all prerequisite courses for this course.
 - **Courses database:** which contains information about all the courses offered by the university. Attributes include the course Id, course name, school, and department.
2. Internal database:
 - **Course option database:** This includes all the possible options that the course can be offered at. It is managed by the dean/head of department who can add, view, remove and modify a course option.

Attributes: Option_id which is an autoincremented id (PK), course_id (FK from external database), day (varchar (10)), time (DateTime), interest_count (incremented when student ticks checkbox and decremented when student unticks checkbox).
 - **Priority database:** This table stores the course id that the student chose for the high priority and the course id that the student chose for the top priority.

Attributes: student_id, high_priority_course_id, top_priority_course_id.
 - **Analysis results database:** this stores the results of the analysis of the course preregistration and the priorities.

Attributes: course_id, interest_count (overall number of students interested in the course regardless of the section), average_priority, course_urgency.



ERD

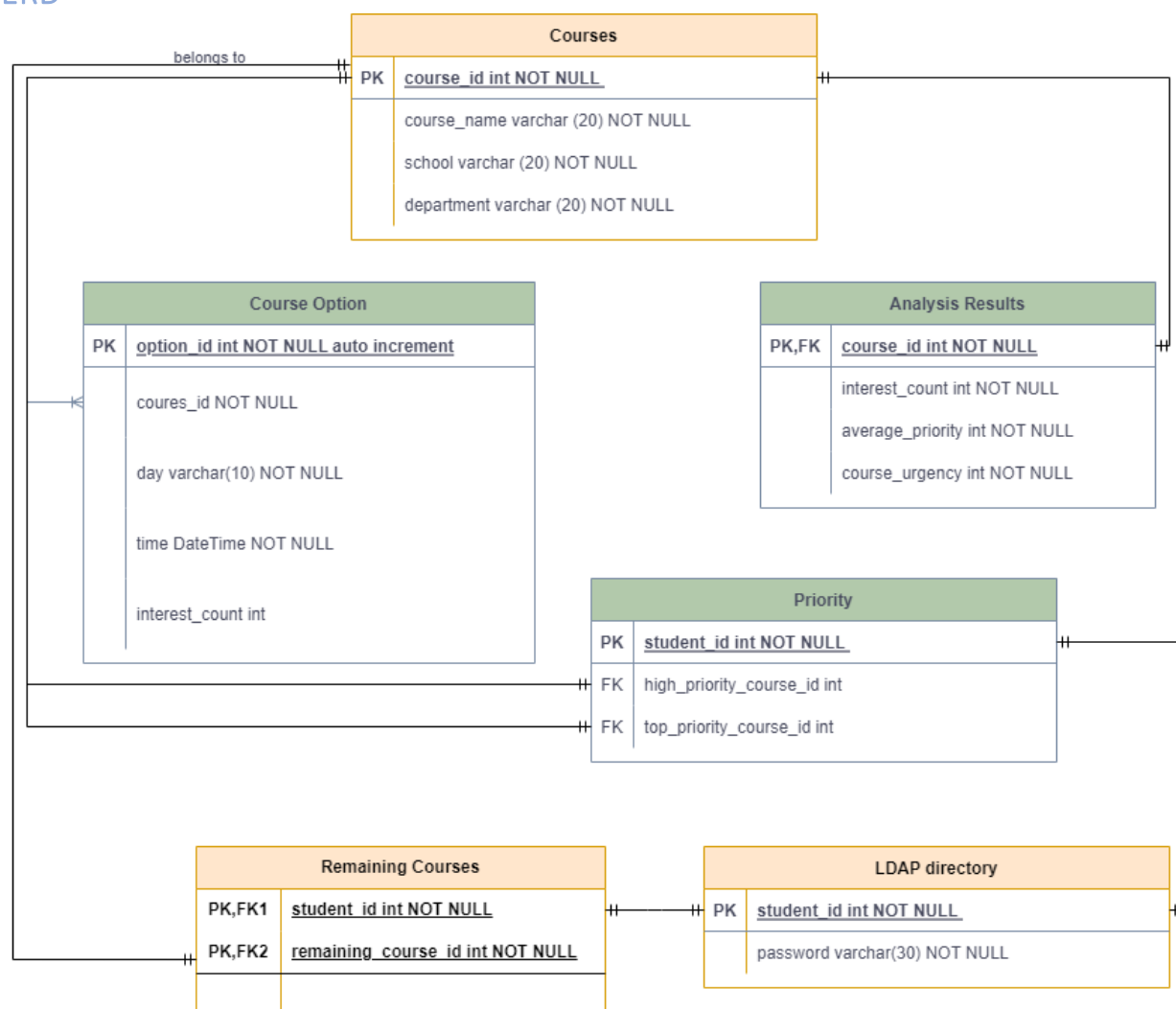


Figure 11: ERD of DRS.

Note: the databases in red are external databases.



Part 6: Teamwork and implementation

Trello board link: <https://trello.com/b/deEtea2y> .

GitHub repo link: <https://github.com/ZainaQudah/Dummy-Registration-System> .

Challenges faced while working in the team.

Collaborating with the team has been an eye-opening experience, despite some initial challenges due to the variety of personalities and backgrounds. During our first meeting, which was on campus, team member (D) wanted to join through Teams (despite her already being there). I thought that this was odd since I believe face to face interactions are crucial, but we reached a compromise by holding the meeting in person while simultaneously starting a meeting on Teams. This ended up being convenient for another team member (B) who had a lecture right after and since it was recorded, I could go back and rewrite our requirement questions better.

Also, during our first meeting, I found it somewhat challenging to refocus the conversation on the main goal of the meeting which is to write the requirements questions rather than being side-tracked by details that were somewhat irrelevant. I remember after I went home from the meeting, I searched how to effectively lead discussion by allowing brainstorming while also knowing when to refocus to ensure time was not wasted. I learnt an effective way to do that was to acknowledge the contribution and then gently reestablish the goal of the meeting. I think this approach is useful, and I plan to use it in other projects as well.

Moreover, working with group member D was occasionally difficult due to her tendency to interrupt. In following meetings, I tried to ensure that everyone was given the opportunity to express their opinions and ideas, resulting in a more inclusive and productive environment.

Team member X didn't attend any of the meetings but was crucial in directing the project pinpointing the tasks that we had to get done (team meeting summaries, front-end implementation, Trello, etc.), they also coordinated the meeting with the stakeholder to answer the requirements questions that we had. This was extremely helpful in understanding the client needs fully, allowing us to create the system and wireframes to match the user and system requirements.

Team member A consistently attended and actively engaged in all our meetings, adding significant value to the discussion of the requirement questions, making it unnecessary for her to be present at a later meeting with one of the stakeholders. She openly communicated to the team that she was overwhelmed with the workload that she had currently and asked for help on the task that was assigned to her (writing meeting summaries). This open and honest communication was critical in ensuring the quality of the requirements, as it allowed us to efficiently balance the effort and prevent overburdening any individual member. I believe that if she had not communicated this issue, it could have resulted in issues such as incomplete or inadequate meeting summaries, which are critical for our project's documentation and future reference. Her proactive attitude not only contributed to the high quality of our team's work, but also generated a supportive team environment.



Team member B constantly attended all meetings, actively engaged in each one, and communicated with great clarity and precision. Her calm and thorough delivery made it easy to understand and focus on the essential points she made, and I intend on being more mindful of my delivery. Her main role was the implementation of the front-end of the functional requirements in reference to the wireframes that we designed as a team.

Team member D consistently attended all meetings, showing a strong commitment to collaboration and team success. This was especially noticeable when team member A required assistance; she instantly stepped in to help, which was critical to the team's continued success. At first, she would interrupt to get her point across, but as we held more and more meetings, she became more mindful, and it was easy for the rest of the team members to really hear what she was saying. She mainly contributed during the wireframing stage, helped us pick a name for the team and created a GitHub repository to upload the implementations to.

I attended all meetings, designed the logo for the team, managed the Trello board of the team, rewrote the functional questions to make them clearer, wrote down the functional and non-functional requirements that we agreed upon in the meeting. To add, when team member D decided to help with the meeting summaries task, I thought that a simple help like providing team member A and D with the notes I wrote during our meeting with the stakeholder, giving them a template for the meeting summaries, labeling the meetings that need to be summaries, and attaching this template in Trello would make their task just a bit easier. I also joined both their summaries into one document tweaking some changes and writing the summary for the last meeting since both didn't have the time to do it.



Description of how multidisciplinary team skills can produce an exceptional solution.

A multidisciplinary team is made up of people with different professions, each with their own set of skills, viewpoints, and knowledge. When I think of our team, I see that it consisted of a project leader (team member C), a front-end developer (team member B), a remote coordinator (team member X), a requirement analyst (team member A), and a UI designer (team member D). This diversity is a strong asset in problem-solving and innovation, and it can lead to extraordinary solutions.

Each team member contributed a distinct skill to the project, each shining at one stage according to their skillset. For example, team member A helped us know what we don't understand about the requirements, and we used this information to write requirement questions to the stakeholder. Team member X coordinated the meeting with the stakeholder which provided us with a clear idea of the functional, non-functional requirements and how the system will look like. Based on the information from this meeting, team member D was able to provide valuable input into the UI design of the system while creating the wireframe. The functional, non-functional requirements and wireframes created as a result were used by team member B who implemented the front end of the system. Team member C kept the timeline ensured by creating a Trello board, coordinated meetings, and directed conversation within the meetings. Directing discussions helped keep focus on the primary goals which is critical in leveraging the various inputs of a diverse team without becoming distracted. This is particularly important in multidisciplinary teams where different perspectives can lead to richer, more well-rounded solutions.

Each discipline brings its own set of approaches to problem-solving. For example, the front-end developer concentrated on how to execute the solution, but the UI designer prioritized usability. Combining these views resulted in more well-rounded and innovative solutions. To add, this problem had numerous dimensions. For example, we are all stakeholders as we are students, so we understand one aspect of the issue. The interview with an instructor helped us gain insight into the system from another perspective.

I believe that since we all come from different backgrounds, we had a different perspective on how to solve the issue at hand, so we encouraged each other to be more creative. (The wireframe and approach to how to solve the issue was a combination of all the perspective and a thorough discussion.)

Moreover, observing how each team member carried themselves, whether positive or negative, helped me reflect on the way I carried myself and ways to improve.

Overall, the team is interlinked, and every team member relies on another member to achieve the best quality overall project.



Part 7: Ethical and social responsibilities.

Identification of ethical and social responsibilities

Ethical and social responsibilities in software engineering cover a variety of ideas and methods to ensure that software development and its outputs respect **user privacy, promote fairness, and benefit society**. A general framework to adhere to ensure ethical and social responsibilities often involves following professional codes such as the ACM Code of Ethics and the IEEE Code of Ethics. These emphasize general moral responsibilities (such as **contributing to society and human well-being, avoiding harm, and being honest**), professional responsibilities (such as **striving for excellence, maintaining confidentiality, and respecting intellectual property**), and leadership principles (such as **leading by example and mentoring younger professionals**). Software engineers face real-world ethical dilemmas, such as **privacy and data protection concerns, intellectual property rights concerns, and deal with unethical requests from clients or employers**. And if software developers don't adhere to a specific framework, unethical software engineering practices can result in **legal ramifications, reputational harm, and detrimental effects on users and society**. Education, training, ethical decision-making frameworks, and the involvement of professional bodies all play a role in promoting ethical practices. The future of software engineering ethics, particularly with developing technologies like AI and machine learning, will create new difficulties that will necessitate informed and considered responses.

Simply put, any software engineer's social obligation is to do all necessary to ensure that the outcomes of their work are of extremely high quality and to presume that it is mission vital, and that someone's life may depend on it being correct. A software engineer must also have a moral code, which means that he or she will not freely and knowingly conduct work that will be misused.

Main ethics principles:

1. **Public:** Software engineers should develop and act in line with the public interest.
2. **Client and employer:** Software engineers must act in the best interests of their customer and employer while also maintaining in the public interest.
3. **Product:** Software engineers are responsible for ensuring that their products and related updates meet the highest professional standards.
4. **Judgment:** Software developers must employ professional judgment while maintaining integrity and independence.
5. **Management:** Software engineering managers and executives must subscribe to and encourage an ethical approach to software development and maintenance management.
6. **Profession:** Software engineers must improve the profession's integrity and reputation in line with public interest.
7. **Colleagues:** Software engineers must be fair and supportive of their coworkers.
8. **Self:** Software engineers should continuously learn about their field and advocate an ethical approach to the practice of their trade.



How ethical and social responsibilities are important to software engineering and software development.

1. **Benefit the public:** A rigid code of ethics ensures that software engineers create products that help people. Without ethics, a software engineer may neglect how their work affects the lives of others or actively develop something destructive. Software should be viewed as a tool that can help our civilization run more smoothly. Without ethics, software engineers may be able to get away with hazardous behavior and decision-making.
2. **Meet professional standards:** Software developers have a different view on ethics and having professional standards set by an ethical code ensures that they are all aligned professionally. Rather than pursuing a goal without first considering its repercussions, employees should ask themselves a few ethical questions before beginning a new project. A software developer working on a new video game, for example, needs to consider how the final product may negatively impact a player's quality of life. Having a professional code also means that software engineers are part of a moral community of like-minded professionals and leaders.
3. **Provide high-quality software:** Software allows banks, communication, power plants, and practically every other modern system to function. Ethics ensures that these systems use high-quality, dependable software that is beneficial and accessible to all. The Code requires software developers to build excellent products that satisfy specifications, pass proper testing, and do not impair quality of life, privacy, or harm the environment.



ACM/IEEE code of ethics

The **IEEE** Code of Ethics describes the ethical and professional standards that its members are expected to meet. It focuses on the following fundamental principles:

1. In professional activities, members must preserve integrity, responsible behavior, and ethical conduct.
2. Put public safety, health, and welfare first, adhere to ethical design and sustainable development methods, preserve privacy, and report potential public or environmental hazards.
3. Increase understanding of traditional and emergent technologies and their societal complications.
4. Avoid real or perceived conflicts of interest (any scenario, transaction, or relationship in which someone's decisions or actions may have a material impact on that person's professional, personal, financial, or business issues) and declare them when someone is aware of them.
5. Don't engage in illegal behavior and reject all sorts of bribes.
6. Offer and accept constructive criticism of technical work, admit and repair errors, acknowledge and credit others' efforts, and make accurate statements based on available data.
7. Maintain and enhance technical proficiency, and only take on assignments if competent.
8. Avoid discrimination and harassment by treating everyone fairly and with respect.
9. Avoid causing harm to others' property, reputation, or employment.
10. To encourage colleagues and coworkers to follow this code of ethics, to work to ensure the code is followed, and not to take revenge against anyone who report a violation.

The **ACM** Code of Ethics and Professional Conduct is a comprehensive framework made to govern computing professionals' ethical behavior. Here's a quick overview of its main components (principles):

1. **General ethical:**
 - Computing professionals should apply their expertise to help society, promote human rights, and reduce the negative effects of computing.
 - Avoid physical or mental harm, the destruction or disclosure of information, and environmental damage.
 - Professionals should be honest, avoid making false claims, and keep their commitments.
 - Be fair and take nondiscriminatory action by encouraging equitable participation and preventing biased discrimination.
 - Professionals should give credit to inventors and uphold intellectual property rights.
 - Protect personal information and individuals' private rights.
 - Keep confidential information private unless it is required by law or the Code.



2. Professional responsibilities:

- Maintain all stakeholders' dignity and respect (professionalism).
- Maintain continuous learning and skill improvement.
- Unless they contradict the ethical standards, follow legal and organizational policies.
- Participate in giving and receiving constructive peer review processes.
- Be impartial and thorough in conducting computer system evaluations, especially when assessing danger.
- Recognize one's own limitations in terms of abilities and competence and only perform work in areas of competence.
- Share technical knowledge and enhance understanding of computing's influence.
- Only use computing resources when authorized and obey data and system access constraints.

3. Professional Leadership:

- In professional computing work, always emphasize the welfare of the public.
- Leaders must guarantee that their organizations fulfill their social responsibilities.
- Emphasize personal and professional growth, worker safety, and well-being.
- Enforce policies that are in accordance with the Code.
- Facilitate educational and skill development opportunities.
- When modifying or retiring systems, minimize the negative impact on users.
- Recognize the increased responsibility that comes with systems becoming a part of society infrastructure (integrated systems).

4. Compliance with the Code:

- Members must follow and promote the principles of the Code.
- Report violations and encourage adherence to the Code.

The ACM and IEEE codes of ethics both emphasize ethical behavior, professional accountability, and public benefit. They both emphasize the significance of preventing injury, whether physical, mental, or environmental, as well as honesty, justice, and non-discrimination. Each code emphasizes the significance of privacy, intellectual property rights, and secrecy. Both foster continuous learning and skill growth, as well as the understanding of one's own limitations in terms of abilities and competence. Both standards also emphasize responsible technology use, such as ethical design and sustainable development. They advocate for the active promotion of these ethical norms throughout the professional community and emphasize the importance of reporting infractions.

The IEEE code is more concerned with practical and professional issues, such as prioritizing public safety, health, and welfare, avoiding conflicts of interest, refusing bribes, and maintaining technical expertise. It also expressly addresses the avoidance of discrimination and harassment, as well as the protection of others' property, reputation, or employment. The ACM code, on the other hand, dives deeper into the larger societal consequences of computer, highlighting the role of computing professionals in advancing human rights and mitigating computing's negative



effects. It emphasizes leadership responsibilities, ensuring that organizations fulfill their social responsibilities, and controlling the societal impact of integrated systems. The ACM code also emphasizes rules for professional leadership as well as leaders' responsibility to implement regulations in compliance with the code.

Reflection of ACM/IEEE on my solution

My project is consistent with several major ethical criteria established by both organizations:

1. **Professional responsibility and public welfare:** The needs and wellbeing of the students and instructors are prioritized by the system. The system is intended to manage schedules efficiently showing a commitment to societal welfare and ethical professional conduct.
2. **Privacy and confidentiality:** The system respects and protects user privacy by utilizing LDAP for secure login and assuring data encryption. This is consistent with the ACM/IEEE principles, which emphasize the significance of safeguarding personal information and protecting privacy rights.
3. **Avoidance of harm:** The system strives to reduce stress and scheduling conflicts. The technology reduces any negative effects on the mental well-being of students by facilitating a smoother registration procedure and minimizing workload imbalance for instructors.
4. **Fairness:** The approach to course scheduling is data-driven and objective, which helps to ensure that all students are treated fairly. Moreover, the priority ensures that courses with low number of students, but high urgency are still taken into advance.
5. Throughout the project, I tried to encourage equitable participation of all team members and various stakeholders.
6. Team members were respectful and supportive of one another, and credit was given where it was due.
7. Team members were aware of their strengths and weaknesses and tasks were distributed accordingly, ensuring competence. Also, constructive criticism was given to the team members while maintaining respect.
8. Testing was done on implementation of the wireframe and functionality ensuring that it meets client standards.



References

Everard, M. (2014). *8 Usability Testing Tools When On A Budget - Usability Geek*. [online] Usability Geek. Available at: <https://usabilitygeek.com/8-usability-testing-tools-when-on-a-budget/>.

LambdaTest. (2022). *30 Top Automation Testing Tools In 2022*. [online] Available at: <https://www.lambdatest.com/blog/automation-testing-tools/>.

www.youtube.com. (n.d.). *What is LDAP and Active Directory ? How LDAP works and what is the structure of LDAP/AD?* [online] Available at: <https://www.youtube.com/watch?v=0FwOcZNjjQA> [Accessed 18 Sep. 2023].

www.linkedin.com. (n.d.). *How do you apply the principle of separation of concerns to avoid the blob anti-pattern in software design?* [online] Available at: [https://www.linkedin.com/advice/0/how-do-you-apply-principle-separation-concerns-1e#:~:text=Separation%20of%20concerns%20\(SoC\)%20is](https://www.linkedin.com/advice/0/how-do-you-apply-principle-separation-concerns-1e#:~:text=Separation%20of%20concerns%20(SoC)%20is) [Accessed 10 Jan. 2024].

www.linkedin.com. (n.d.). *Jquery vs React: Difference between jQuery and React [2023]*. [online] Available at: <https://www.linkedin.com/pulse/jquery-vs-react-difference-between-2023-khadarvalli/> [Accessed 10 Jan. 2024].

Koutsonikola, V. and Vakali, A. (2004). LDAP: framework, practices, and trends. *IEEE Internet Computing*, 8(5), pp.66–72. doi:<https://doi.org/10.1109/mic.2004.44>.

www.linkedin.com. (n.d.). *How do you secure your system with programming languages and frameworks?* [online] Available at: <https://www.linkedin.com/advice/0/how-do-you-secure-your-system-programming-languages> [Accessed 10 Jan. 2024].

TMS. (2018). *14 Best Web Development IDE in 2019 [CSS, HTML, JavaScript] | TMS*. [online] Available at: <https://tms-outsource.com/blog/posts/web-development-ide/>.

Cook, B. (2022). *Everything to Know About Software Engineering Ethics*. [online] Fellow.app. Available at: <https://fellow.app/blog/engineering/engineering-everything-you-need-to-know-about-software-engineering-ethics/>.

Quora. (2015). *What is the additional social responsibility of software engineers?* [online] Available at: <https://www.quora.com/What-is-the-additional-social-responsibility-of-software-engineers> [Accessed 10 Jan. 2024].

Anon, (n.d.). *Continuing Education for Professional Engineers PDH-PRO» Ethics in Software Engineering: A Key Component of Professional Practice*. [online] Available at: <https://pdh-pro.com/pe-resources/ethics-in-software-engineering/>.