

*SOFTWARE ENGINEERING & CONCEPTS – LAB MANUAL*

***Automated Exam Time-Table Generator***

*Sneha Sajeevan 220701281*

*Suranjana S 220701293*

*Shree Harshini S 220701269*

*Shubalekha G 220701275*

*Sriniranjan V 220701287*

[Overview of the Project 2](#_gjdgxs)

[Business Architecture Diagram 3](#_30j0zll)

[Requirements as User Stories 4](#_1fob9te)

[Architecture Diagram depicting the 5](#_3znysh7)

[Test Strategy 6](#_2et92p0)

[Deployment Architecture of the application 7](#_tyjcwt)

# Overview of the Project

Through this project we try to address the pressing issue of the exam timetable creation which provides a suboptimal schedule that negatively impacts the students’ well being and academic performance. The current methods used to create exam time tables don't consider crucial factors such as providing an optimal schedule that is not prone to errors and takes the students' health (mental and physical) into account. This can lead to scenarios where exams are clustered closely together which might lead to delivering an inconsistent schedule and potentially affecting the students performance.

To understand this problem on a deeper level we conducted a survey among 70 different students to understand the problem which exists with the current exam schedule. The responses showed the following results:

1. More than 36% felt that the exam schedule has affected their stress level extremely negatively.
2. More than 56% of the students said that the exam schedule affects their sleep schedule negatively.
3. Only 3% of the students who attended the survey voted that the exam schedule also does not affect their physical health negatively.
4. Only more than 35% percent do not feel ready for the exams.
5. 40 out of 70 student felt that the current exam schedule is very stressful and gives them anxiety
6. 35 out of 70 students suggested that the holidays between the exams should be effectively distributed.

The solution that we proposed will offer several benefits to the users in terms of accuracy, flexibility, transparency, collaboration and accessibility. As the accuracy in exam scheduling increases the students and the faculties will be benefitted. The sophisticated algorithm minimizes the conflicts, optimizes the resource allocation and ensures fair distribution of exams and in turn reduces the likelihood of errors and conflicts in the timetable and provides a reliable schedule that users can trust. The flexibility is ensured as the diverse needs of both the students and faculties are aligned with the generated timetable. Users would have real-time access to updates and notifications regarding any changes or updates. Transparency fosters trust and confidence among the stakeholders, as they can easily track the status of exams, venues and scheduling decisions. Users would have access to the exam timetable anytime in preferred devices. The system would also facilitate collaboration between different stakeholders to easily coordinate, share resources and resolve scheduling conflicts. It also allows students to engage in the process of providing feedback and suggesting improvements.





# 

# Business Architecture Diagram

Business Need:

The business need for the project is to develop an automated exam timetable generator to streamline the process of scheduling exams for educational institutions. This system aims to address challenges such as manual scheduling errors, inefficiencies in resource allocation, and the need for personalized scheduling to accommodate diverse student needs.

Current Process:

Currently, the exam scheduling process may vary across institutions. In some cases, it's a manual process where administrators or faculty members manually create exam timetables based on factors like course schedules, venue availability, and faculty availability. In other cases, institutions may use basic automated tools or spreadsheets for scheduling, but these tools may lack advanced features for optimization and customization.

Current Process for Different Personas:

- Administrators: Administrators oversee the entire exam scheduling process, from setting up exam periods to allocating resources. They may use spreadsheets or scheduling software to manage exam timetables, often facing challenges in coordinating schedules, avoiding conflicts, and ensuring fairness.

- Faculty Members: Faculty members provide input on exam dates and preferences but may have limited visibility into the overall scheduling process. They rely on administrators to finalize exam schedules and may encounter issues with conflicting schedules or inadequate preparation time.

- Students: Students rely heavily on exam timetables to plan their study schedules and prepare for exams. They often face challenges such as overlapping exam times, insufficient study time between exams, or conflicts with personal commitments.

Business Problems:

1. Manual Errors: The current manual or semi-automated process is prone to errors, leading to scheduling conflicts, inaccurate resource allocation, and dissatisfaction among stakeholders.

2. Inefficiency: The lack of automation results in inefficiencies in the scheduling process, consuming valuable time and resources of administrators and faculty members.

3. Lack of Personalization: The current process may not adequately account for the diverse needs of students, leading to suboptimal exam schedules that may impact student performance and well-being.

4. Limited Visibility: Different personas involved in the process may have limited visibility into the overall scheduling process, leading to miscommunication, misunderstandings, and delays in decision-making.

# User Stories:

E: Automated Exam Time-Table Generator

F: Exam schedule management

U: As a administrator i want to include exam details such as subjects, date range and exam slots

AC: Start and end dates should be clearly mentioned and unique identifiers should be attached with the subject. It should validate the inputs and provide an intuitive user interface.

T: Create a form to input exam details.

T: Implement validation checks for exam details.

F: Constraints handling.

U: As an administrator I want the system to handle constraints such as room availability and student preference.

AC: Ensure that the system dynamically adjusts scheduling based on real time availability and fairness and efficiency is maintained.

T: develop algorithms to handle room and faculty availability.

T: implement a module to collect and incorporate student preferences.

E: User Access and Authentication

F: User Registration and Authentication

U: As a student or administrator, I want to register an account and login securely to access the system

AC: Provide user registration features allowing the users to create an account securely and implement a good login mechanism with password encryption and hashing.

T: Design a user registration forms for both the students and administrators

T: Implement authentication using secure protocols.

F: Role-based Access Control

U: As an administrator I want to assign roles and permissions to users.

AC: The administrator can easily modify roles and permissions as needed. The system should ensure that users can access functionality and data relevant to their assigned roles.

T: Develop role-based access control mechanisms

T: Create an interface for administrator to manage the user roles

E: Exam Schedule viewing and modifications

F: Exam schedule display

U: As a student or administrator, i want to view the exam schedule in a user friendly format

AC: The exam schedule should be accessed effortlessly through a user friendly interface and should be presented in a clear and intuitive format. The system should also provide filter and search functions for easy navigation

T: Design a dashboards to display the exam schedule

T: Implement filter and search functions

U: As an administrator, I want to modify the exam schedule if needed

AC: Administrator should be able to modify the time table with appropriate permission and the changes should be logged and communicated to the relevant stakeholders. They should be able to adjust the exam date, time and location.

T: Develop a module to edit exam details

T: Implement audit logging to track schedule modifications

E: Communication and Notifications

F: Notification

U: As a student or administrator, I want to receive notifications about exam schedule changes or announcements

AC: The users should receive timely notification via email. The notification should have clear information about the changes in the schedule.

T: Implement email notifications

T: Develop a notification system for in-app alerts

F: Communication

U: As an administrator, i want to communicate important information to students regarding exams

AC: Information given should be clear and easily understandable by students. The information should include all relevant details such as exam dates, times and location and the communication should be distributed to all students via mail

T: create a messaging system for administrators to send announcement

T: Implement features for students to ask questions or request clarifications

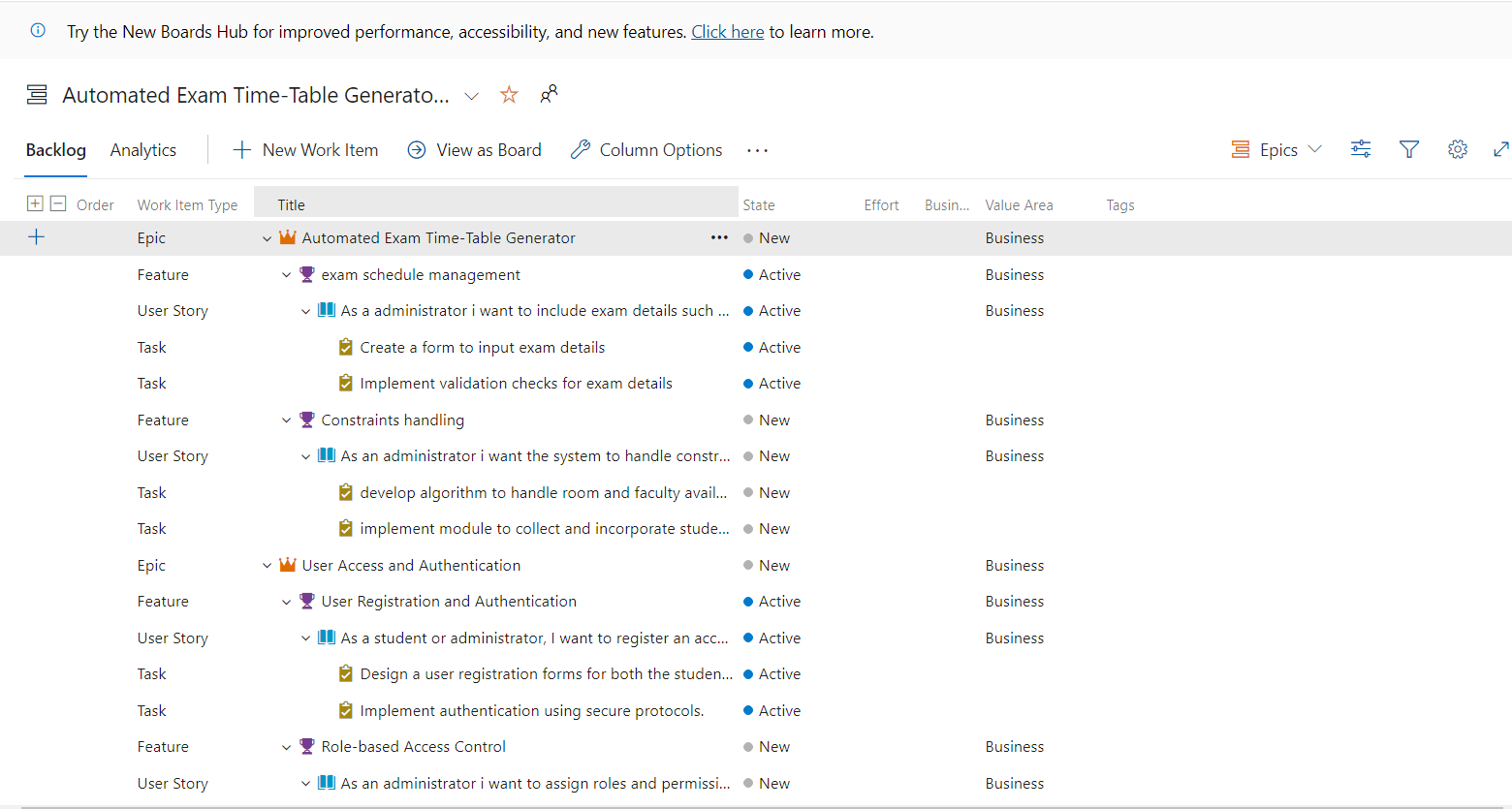
F: Feedback collection

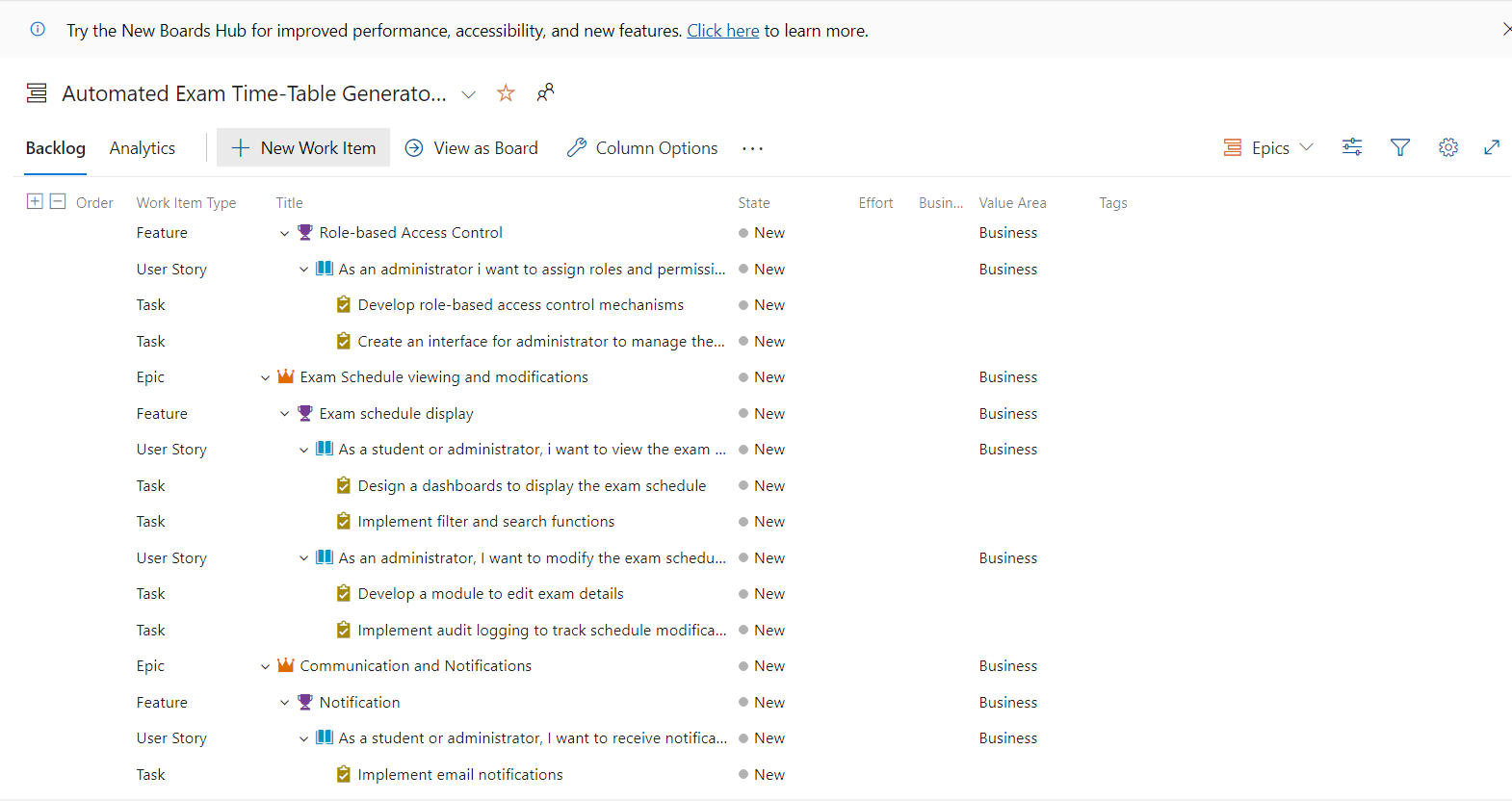
U: As a student, I want to provide feedback on the exam schedule and process

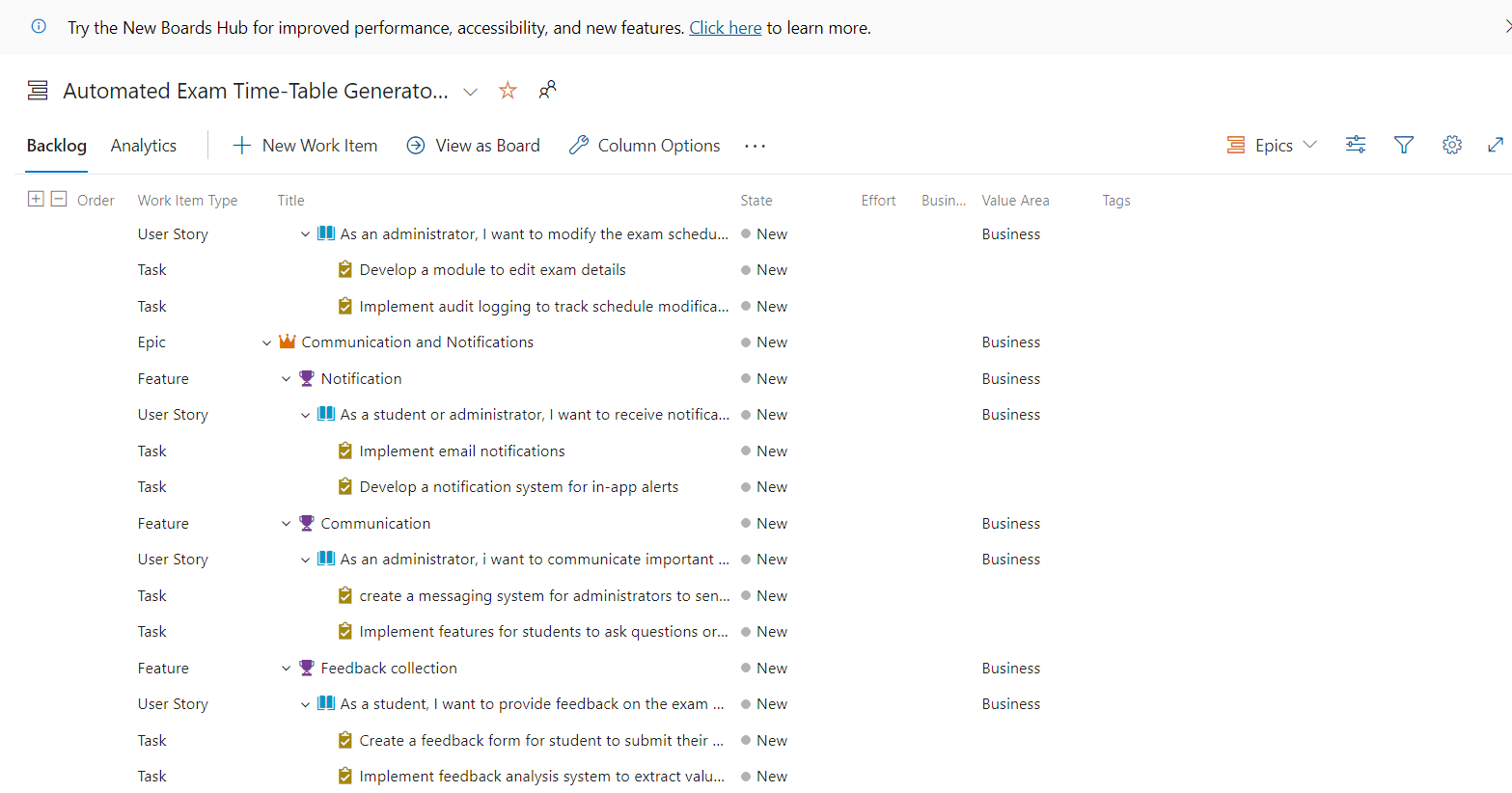
AC: Submission of feedback should be easily accessible to all students and should allow students to express their opinions anonymously.

T: Create a feedback form for student to submit their opinion

T: Implement feedback analysis system to extract valuable insights







NON FUNCTIONAL REQUIREMENTS:

## Non-Functional Requirements for Exam Timetable Generator Project

Non-functional requirements (NFRs) define the overall characteristics of a system beyond its specific functionalities. Here are some key NFRs for an exam timetable generator project:

**1. Performance:**

* **Response Time:** The system should generate timetables within a reasonable timeframe. Aim for timetables to be generated in seconds or minutes, depending on the project size and complexity.
* **Scalability:** The system should be able to handle an increasing number of users and exams without significant performance degradation. Consider scalability for future growth.
* **Availability:** The system should be available to users most of the time. Define an uptime target (e.g., 99.5%) and a recovery time objective (RTO) for unplanned outages.

**2. Usability:**

* **User Interface:** The user interface should be intuitive and easy to use, regardless of technical expertise.
* **Data Input:** The system should provide a clear and streamlined way for users to enter exam details. Consider validation and error handling to prevent incorrect data entry.
* **Timetable Presentation:** The generated timetable should be clearly presented and easy to understand. Export options (e.g., PDF, CSV) might be useful.

**3. Reliability:**

* **Accuracy:** The generated timetable should be accurate and free of conflicts (e.g., no student taking multiple exams at once, room availability conflicts).
* **Data Integrity:** The system should ensure the accuracy and consistency of exam data stored in the database. Implement mechanisms to prevent data corruption.
* **Error Handling:** The system should handle unexpected errors gracefully, provide informative error messages to users, and attempt to recover from errors whenever possible.

**4. Security:**

* **Authentication and Authorization:** The system should restrict access based on user roles (e.g., only authorized users can create or edit timetables). Implement secure authentication mechanisms.
* **Data Security:** Exam data, especially student information, should be protected from unauthorized access, modification, or deletion. Encryption might be necessary for sensitive data.
* **Backup and Recovery:** The system should have a regular backup strategy to ensure data recovery in case of system failures or security incidents.

**5. Maintainability:**

* **Modular Design:** The system should be designed with modular components to facilitate future modifications and upgrades.
* **Documentation:** Clear and well-maintained documentation should be available for developers and administrators to understand the system's functionality and configuration.
* **Logging and Monitoring:** The system should log relevant events for troubleshooting and performance analysis purposes.

POKER PLANNING METHADOLOGY:

Poker planning methodology, while not a direct fit for a single developer project like an exam timetable generator, can be conceptually adapted to estimate the effort involved in developing the project. Here's how you can adapt the idea:

**Adapting Poker Planning:**

1. **Break Down Work Items:** Divide the exam timetable generator project into smaller, well-defined tasks. Examples might include:
   * Designing the user interface for exam data input.
   * Implementing algorithms for generating conflict-free timetables.
   * Integrating with a database for storing exam data.
   * Developing functionalities for user management and access control (if applicable).
2. **Define Estimation Scale:** Instead of using physical poker cards, create a custom estimation scale relevant to your development process. This could be:
   * **Effort Points (e.g., 1, 2, 3, 5, 8):** Assign point values based on the estimated effort for each task.
   * **T-Shirt Sizes (XS, S, M, L, XL):** Use relative sizes to represent the perceived complexity of each task.
   * **Ideal Days (1, 2, 4, 8, 16):** Estimate the ideal number of days a skilled developer might take to complete each task.
3. **Estimation Session:** Gather all developers involved in the project (even if it's just you) and go through each task.
   * Briefly explain the task and its requirements.
   * Each developer secretly estimates the effort using the chosen scale.
   * Reveal estimations simultaneously and discuss any significant discrepancies.
   * Through discussion, arrive at a consensus effort estimate for each task.

**Benefits of Adapting Poker Planning:**

* **Improved Effort Visibility:** This process helps you gain a better understanding of the overall effort required for the project and identify potential challenges.
* **Enhanced Communication:** The discussion phase promotes communication and fosters a collaborative approach to project planning.
* **Realistic Expectations:** By estimating effort, you can set more realistic deadlines and resource allocation plans.

# 

# Architecture Diagram

# MVC ARCHITECTURE DIAGRAM:

* **Model:** This represents the data layer of the application. It encapsulates the data structures and logic related to the exam timetable. Here are some possible Model components for the timetable generator:
  + Classes representing Courses, Exams, Students, Teachers, Rooms, etc.
  + Functions for accessing and manipulating the timetable data (e.g., adding exams, checking for conflicts).
  + Connection to a database (optional) for storing persistent data.
* **View:** This represents the user interface of the application. It displays information to the user and allows them to interact with the system. Here are some examples of View components in the timetable generator:
  + User interface elements for entering exam details (course, date, duration, etc.)
  + A table or calendar displaying the generated exam timetable
  + Buttons for generating the timetable, saving results, etc.
* **Controller:** This acts as an intermediary between the View and the Model. It receives user input from the View, processes it using the Model's logic, and updates the View accordingly. Here are some functionalities of the Controller in the timetable generator:
  + Receives user input for exam details
  + Calls the Model to generate the exam timetable considering constraints (e.g., no conflicts, room availability)
  + Passes the generated timetable data to the View for display
  + Handles errors and exceptions

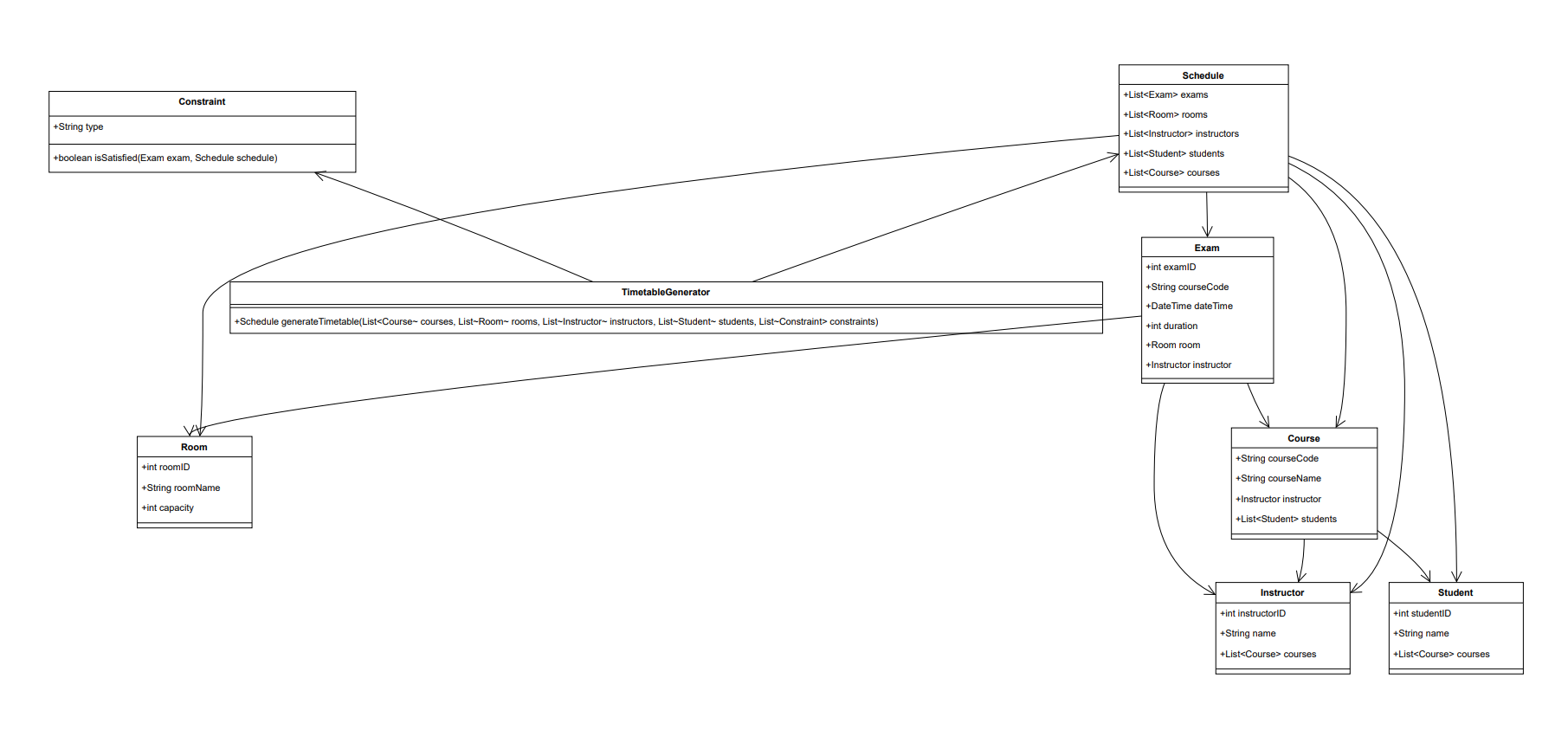
This is a simplified MVC architecture for the exam timetable generator project. The specific implementation details will depend on the chosen programming language, framework, and desired functionalities.

# Design principle used:

**Heuristics:**

* **Strengths:**
  + Can significantly improve search efficiency by guiding towards promising solutions.
  + Offer flexibility to prioritize specific goals (e.g., minimizing student conflicts).
* **Considerations:**
  + The effectiveness of heuristics depends on the specific problem and data.
  + Poorly designed heuristics can lead to suboptimal solutions or even prevent finding a feasible solution.

# Class Diagram:



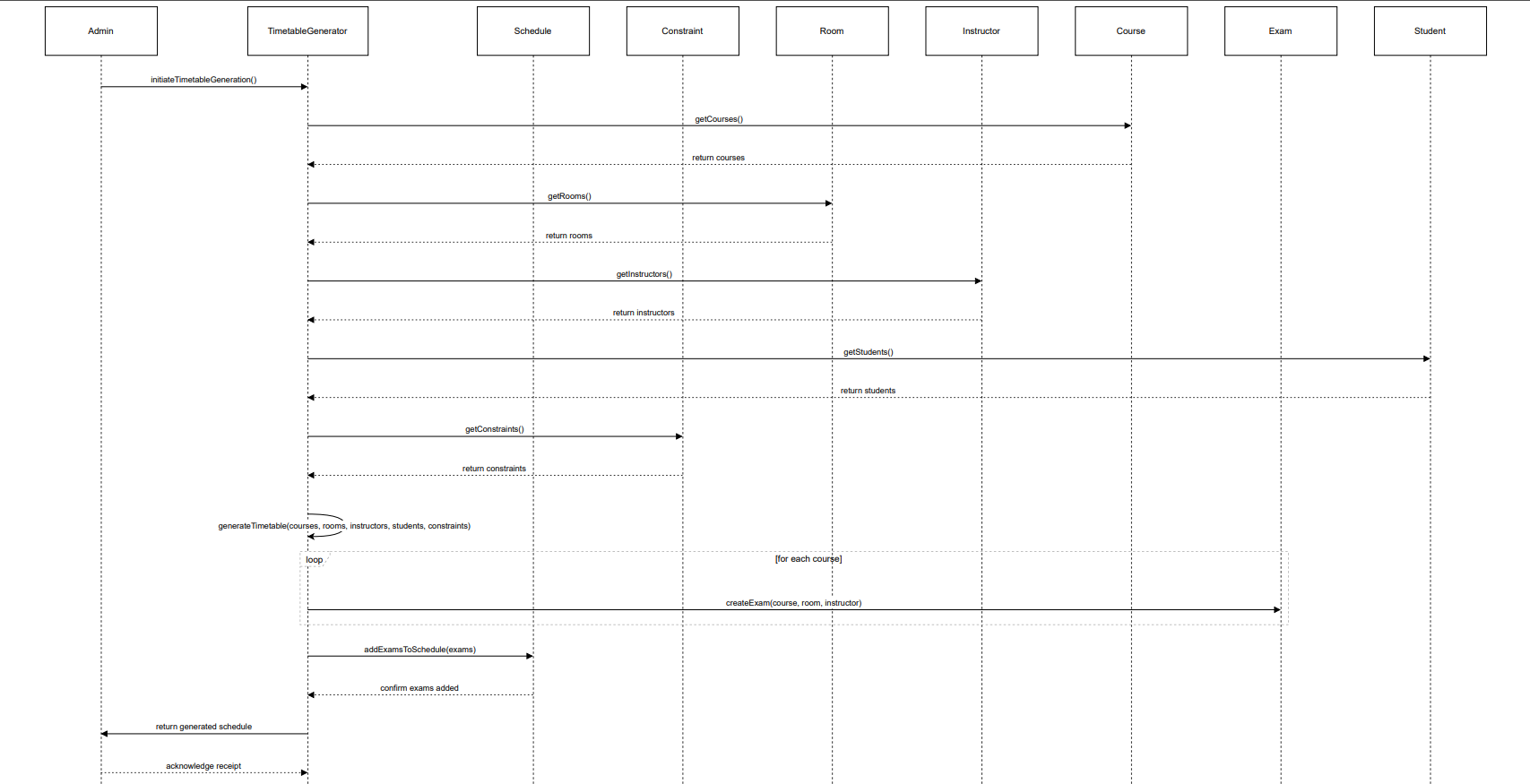
* **Schedule** : This class likely represents the generated exam timetable. It might contain attributes specifying the time slots and the exams assigned to those slots. However, the details of the attributes are not shown in the diagram.
* **Constraint** : This class likely represents constraints that the timetable generator considers when creating the schedule. There are no attributes shown for this class, but it might include things like student availability, room capacity limitations, or teacher conflicts.
* **Exam** : This class represents an exam. It has attributes like examId, courseId (which references the Course class), date, startTime, and duration.
* **Room** : This class represents a room where exams can be held. It has attributes like roomId, roomCapacity, and location.
* **Course** : This class represents a course for which an exam is conducted. It has an attribute courseCode.
* **SingCode** : The purpose of this class is not entirely clear from the diagram. It might be a typo or a specific detail related to the system's implementation.
* **Student** : This class represents a student taking the exam. It has attributes like studentId and studentName.
* **Being** : The purpose of this class is not entirely clear from the diagram. It might be a base class for Student and Teacher or might serve another purpose in the system.

The diagram shows the following relationships between the classes:

* **Schedule** is likely created based on **Exams** and adheres to the defined **Constraints**.
* An **Exam** is related to a specific **Course**.
* An **Exam** can be scheduled in a particular **Room**.
* A **Course** has a unique course code (courseCode).
* Students (Student) and potentially Teachers (represented by another class but might inherit from Being) take **Exams**.

Overall, the class diagram provides a basic outline of the data model for the exam timetable generator system. It illustrates how the system stores information about exams, courses, rooms, and students, and how these elements influence the generation of the exam timetable.

# Sequence Diagram:



**Sequence Diagrams in Exam Timetable Generator Projects**

Sequence diagrams depict the message flow between objects involved in a specific scenario. In an exam timetable generator project, a sequence diagram might illustrate how the system generates an exam timetable upon receiving user input. Here's a breakdown of the typical elements:

* **Participants:** These represent the objects involved in the interaction. They can include:
  + User: The person interacting with the system to generate the timetable.
  + TimetableGenerator: The main object responsible for generating the timetable based on constraints and user input.
  + ConstraintChecker (Optional): An optional object that verifies if the generated timetable adheres to defined constraints (e.g., no student conflicts, room limitations).
* **Messages:** Arrows represent messages exchanged between participants. These messages indicate actions or requests being sent.

**General Flow of a Sequence Diagram:**

1. **User Initiation:** The user activates the system by providing exam details like course list, exam duration, and other relevant information.
2. **Timetable Generation:** The TimetableGenerator receives the user input and starts generating the timetable.
3. **Constraint Validation (Optional):** The TimetableGenerator might interact with a ConstraintChecker to validate the generated timetable against predefined constraints.
   * **Constraints Satisfied:** If the timetable adheres to all constraints, it's presented to the user.
   * **Constraints Violated:** If constraints are violated, the TimetableGenerator refines the timetable to address the issues. This might involve looping through generation and validation until a feasible solution is found. The ConstraintChecker might be involved in each iteration (optional).
4. **User Modification (Optional):** The user might request changes to the initial timetable. The system interacts with the user to understand the modifications, and the TimetableGenerator adjusts the timetable accordingly.
5. **Timetable Presentation:** Finally, the generated timetable is presented to the user. It might be optimal or adjusted based on constraints.
6. **System Deactivation:** Once the user acknowledges the timetable, the interaction ends, and the system deactivates.

# Test Strategy

Test Plans:

1. User Interface Testing: Ensure the user interface is intuitive and responsive across different devices and browsers.

2. Functionality Testing: Validate that all features, including exam scheduling, user authentication, and data storage, function as expected.

3. Integration Testing: Verify interactions between different modules, such as the scheduler, database, and user interface.

4. Performance Testing: Assess system performance under varying loads to ensure scalability and responsiveness.

5. Security Testing: Test for vulnerabilities in user authentication, data encryption, and access control mechanisms.

Test Cases:

1. Input Exam Dates

- Happy Path: Administrator inputs valid exam dates.

- Error Scenario: Administrator inputs invalid or overlapping exam dates.

2. Generate Exam Timetable

- Happy Path: Scheduler generates a complete and conflict-free exam timetable.

- Error Scenario: Scheduler fails to generate timetable due to conflicting schedules or resource constraints.

3. View Exam Timetable

- Happy Path: Students and faculty can view their respective exam timetables.

- Error Scenario: Students or faculty encounter errors while accessing the timetable due to technical issues.

4. Update Exam Schedule

- Happy Path: Administrator successfully updates exam schedule with revised dates or times.

- Error Scenario: Update fails due to database errors or invalid input.

5. Handle Exception

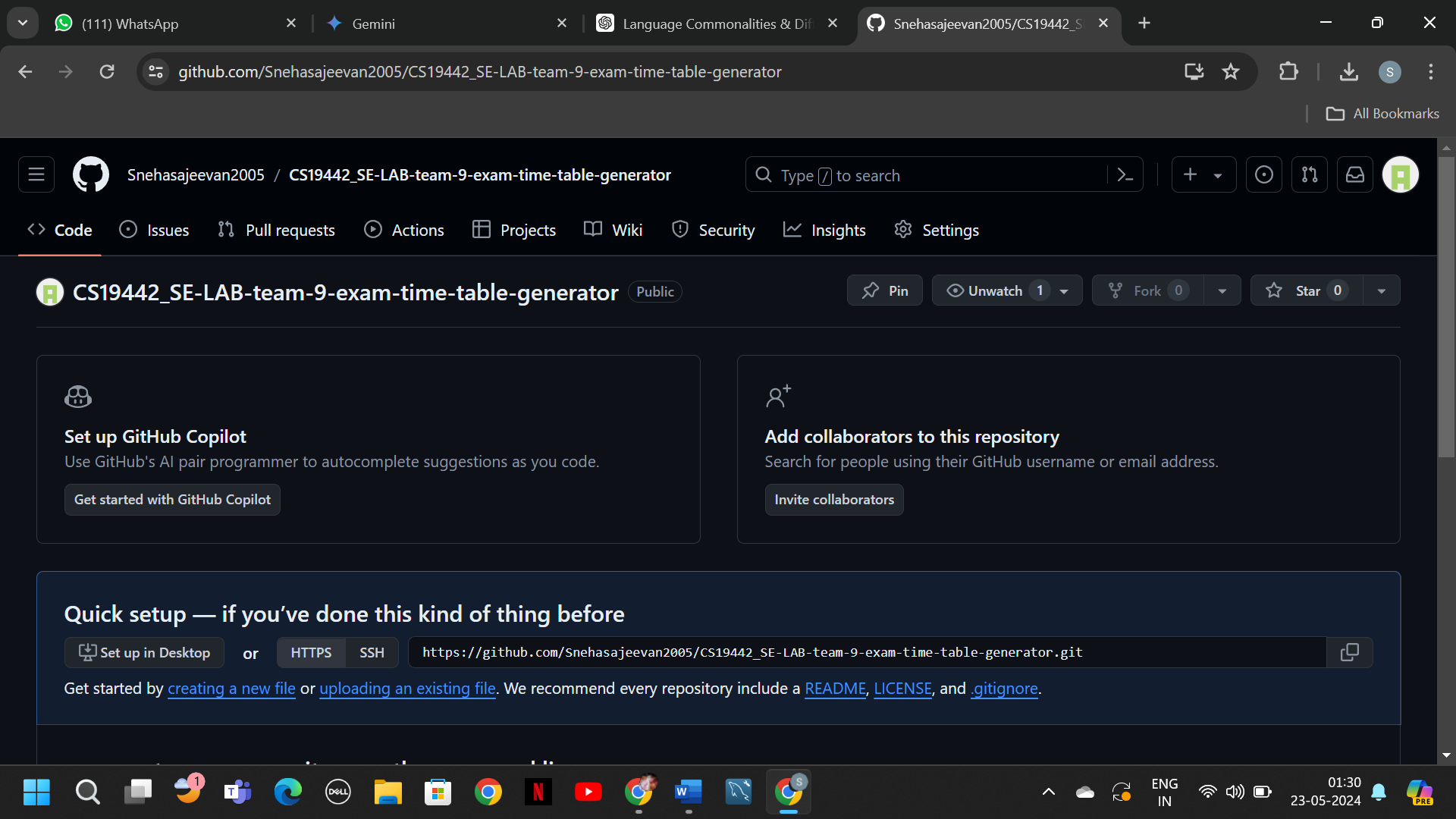
- Happy Path: System gracefully handles exceptions and provides meaningful error messages.

- Error Scenario: System crashes or displays cryptic error messages during unexpected situations.

GitHub Repository Structure and Naming Conventions:

- Project Structure: The repository follows a modular structure, with separate directories for frontend, backend, tests, and documentation.

- Naming Conventions: Files and directories are named descriptively, following camelCase for JavaScript files, snake\_case for Python files, and kebab-case for CSS and HTML files.



DevOps Architecture and Tools Used in Azure:

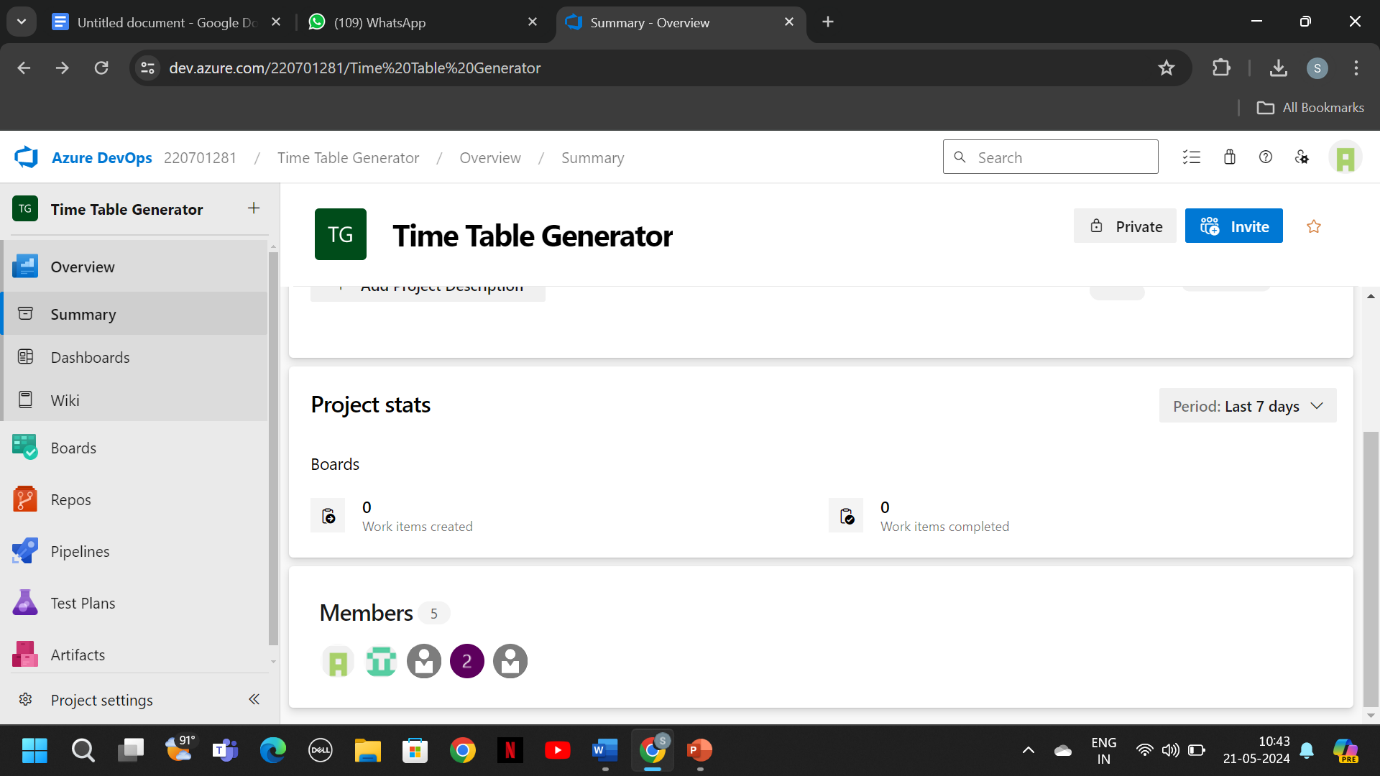
- Continuous Integration/Continuous Deployment (CI/CD): Utilizes Azure Pipelines for automated builds and deployments.

- Version Control: GitHub repository serves as the central version control system.

- Issue Tracking: Azure Boards tracks project tasks, bugs, and user stories.

- Collaboration: Azure Repos and GitHub facilitate collaboration among team members, enabling seamless code reviews and pull requests.

- Monitoring and Logging: Azure Monitor captures telemetry data for monitoring system performance and logs for debugging purposes.



# Deployment Architecture of the application

Frontend Application:

The user interface is accessible via web browsers or mobile devices.

Built using HTML, CSS, and JavaScript frameworks like React or Angular.

Hosted on a web server such as Apache or Nginx.

Backend Services:

Responsible for handling business logic, data processing, and interactions with the database.

Developed using programming languages like Python, Java, or Node.js.

Utilizes frameworks such as Flask, Django, Spring Boot, or Express.js.

Database Server:

Stores and manages data related to exams, users, schedules, and resources.

Can be a relational database (e.g., MySQL, PostgreSQL) or a NoSQL database (e.g., MongoDB, Cassandra).

Hosted on a dedicated database server or a cloud-based database service like Azure SQL Database or Amazon RDS.

Authentication and Authorization Services:

Manages user authentication and authorization, ensuring secure access to the application.

May use OAuth, JWT tokens, or other authentication protocols.

Can be implemented using dedicated identity management services like Auth0 or custom authentication logic.

Load Balancer and Reverse Proxy:

Distributes incoming traffic across multiple instances of the frontend and backend services to ensure scalability and reliability.

Acts as a reverse proxy to handle SSL termination, caching, and routing requests to the appropriate backend service.

Examples include Nginx, HAProxy, or cloud-based load balancing services like Azure Load Balancer or AWS Elastic Load Balancing.

Containerization and Orchestration:

Uses containerization technology like Docker to package applications and their dependencies into portable containers.

Orchestration tools like Kubernetes or Docker Swarm manage containerized applications, scaling instances based on demand and ensuring high availability.

Deployment Flow:

Code Deployment:

Developers push code changes to the version control repository (e.g., GitHub).

Continuous integration/continuous deployment (CI/CD) pipelines trigger automated builds and tests.

Containerization:

Docker containers are built for the frontend and backend services, along with any necessary dependencies.

Orchestration:

Kubernetes or similar orchestration tools deploy containerized applications to a cluster of nodes, ensuring fault tolerance and scalability.

Load Balancing and Routing:

Traffic is routed through a load balancer or reverse proxy, which distributes requests to available instances of the frontend and backend services.

Database Connection:

Backend services establish connections to the database server to read and write data.

Authentication and Authorization:

User authentication and authorization services validate user credentials and provide access tokens for authenticated users.

Monitoring and Logging:

Monitoring tools like Prometheus, Grafana, or Azure Monitor track application performance, resource utilization, and user activity.

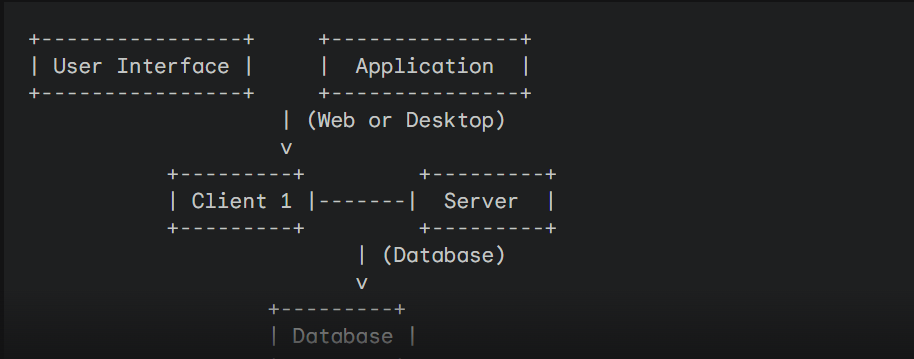
Logs generated by the application are aggregated and analyzed for debugging and troubleshooting.

Deployment Environment:

The application can be deployed on-premises within an organization's data center or in the cloud using platforms like Microsoft Azure, Amazon Web Services (AWS), or Google Cloud Platform (GCP).

Cloud-based deployment offers advantages such as scalability, elasticity, and built-in services for managing infrastructure, databases, and security.

Deployment architecture can vary based on factors like scalability requirements, budget constraints, and compliance regulations.



 **User Interface:** This represents the user interface through which users interact with the system. It can be a web application accessible through a web browser or a desktop application installed on the user's device.

 **Application Server:** This server combines the application logic and the database functionality. It houses the code responsible for generating the timetable, processing user input, and storing and retrieving data. This simplified approach is suitable for smaller projects.

 **Client N:** These represent the various users (e.g., administrators, faculty) accessing the system through their web browsers or using the installed desktop application.

 **Database:** This stores all the exam data, course information, and potentially user data (if applicable). The application server directly accesses and manages the database in this configuration.