Components of the ppt:

**Components of the ppt:**

* Heading
* Introduction
* Phases of development
* Tech-stacks used softwares used and imp links
* Components of project
* 1 database connectivity 2 genetic algo 3 future scope 4 development
* Future scope
* Conclusion (usages, target audience, self made project)
* Our team
* Thanks slide

Introduction

### **Slide 2: Introduction**

#### **Overview:**

The Timetable Generator project automates the creation of timetables for educational institutions using a genetic algorithm. The goal is to efficiently schedule classes, reduce errors and save administrative time.

#### **Problem Statement:**

Manually scheduling timetables is time-consuming and prone to errors. Conflicts in instructor availability, room assignments, and course duration can lead to inefficiency. This project addresses these challenges by using a genetic algorithm to generate conflict-free schedules.

Phases of dev

### **Slide 3: Phases of Development - Overview**

#### **Phases of Development:**

1. **Phase 1**: Research and Planning
2. **Phase 2**: Database Setup
3. **Phase 3**: Genetic Algorithm Implementation
4. **Phase 4**: UI/UX Design
5. **Phase 5**: Testing and Refinement

Tech stacks used and etc

### **Slide 9: Tech Stacks, Software, and Important Links**

#### **Tech Stack:**

* **Programming Languages**: Python
* **Libraries**:
  + **Streamlit**: For UI development.
  + **SQLite**: For database management.
  + **Genetic Algorithm Libraries**: Custom-built algorithm for timetable generation.

#### **Software Used:**

* **Google Colab**: For running the genetic algorithm.
* **Eclipse IDE**: For coding and testing locally.
* **SQLite**: For database management.

#### **Important Links:**

* GitHub repository (if available) for the project code and documentation.
* Articles or research papers on genetic algorithms used in scheduling.

components of project

### **Slide 10: Components of the Project**

#### **1. Database Connectivity**

* The SQLite database stores course, instructor, and timetable data.
* Tables for courses, instructors, and schedules ensure data integrity.
* The db\_setup.py script handles database setup and data retrieval.

#### **2. Genetic Algorithm**

* Optimizes timetable generation by evolving a population of candidate schedules.
* Fitness function evaluates timetables based on conflict resolution, instructor availability, and room assignments.

#### **3. Future Scope**

* Advanced features like room capacity, personalized schedules for students, and integration with campus systems.

#### **4. Development Challenges**

* Ensuring no conflicts in schedules and refining the algorithm for better performance.

4 phases dev

### **Slide 4: Phase 1 - Research and Planning**

#### **Key Activities:**

* **Research**: Explored genetic algorithms and their application in scheduling problems.
* **Understanding Constraints**: Identified the main constraints for timetables (e.g., instructor availability, room assignments, course durations).
* **Requirement Gathering**: Collected information from educational institutions to understand their needs, such as courses, instructors, and class schedules.
* **Project Planning**: Defined the project scope, divided tasks into phases, and set a timeline for development.

### **Slide 5: Phase 2 - Database Setup**

#### **Key Activities:**

* **Database Design**:
  + Designed the relational database schema for courses, instructors, and timetables.
  + Created tables for storing course details, instructor availability, and scheduling data.
* **Tables and Relationships**:
  + **Courses Table**: Stores course names, timings, and instructor details.
  + **Instructors Table**: Stores instructor names, their available hours, and courses they teach.
  + **Schedule Table**: Stores the final timetable, mapping instructors, courses, and rooms to specific time slots.
* **Implementation**:
  + Set up SQLite for local storage.
  + Created a Python script (db\_setup.py) for initializing and managing the database.

### **Slide 6: Phase 3 - Genetic Algorithm Implementation**

#### **Key Activities:**

* **Genetic Algorithm Overview**:
  + The genetic algorithm is used to optimize the schedule by generating multiple possible solutions (timetables) and evaluating them based on a fitness function.
  + **Fitness Function**: Measures how well the timetable satisfies the constraints (e.g., no overlapping classes, instructor availability).
* **Steps in the Algorithm**:
  + **Population Initialization**: Generate an initial population of random timetables.
  + **Selection**: Evaluate the fitness of each timetable and select the best-performing ones.
  + **Crossover and Mutation**: Combine selected timetables (crossover) and introduce small random changes (mutation) to generate new timetables.
  + **Termination**: Continue evolving the population until an optimal or near-optimal solution is found.
* **Key File**: genetic\_algorithm.py.

### **Slide 7: Phase 4 - UI/UX Design**

#### **Key Activities:**

* **Streamlit Interface**:
  + Developed a user interface using Streamlit to allow administrators to view and manage the generated timetable.
  + **User Features**:
    - Display timetable results with options to filter by course or instructor.
    - Option to regenerate timetables based on new inputs.
    - Admins can add or modify course and instructor details via the interface.
* **UI Design Considerations**:
  + **Simplicity**: Ensured the UI is simple and intuitive for administrative users.
  + **Interactivity**: Allowed real-time interaction with the generated timetables (e.g., viewing different courses or instructors).
* **Key File**: display\_schedule.py.

### **Slide 8: Phase 5 - Testing and Refinement**

#### **Key Activities:**

* **Testing the Algorithm**:
  + Tested the genetic algorithm on different datasets to ensure it generates valid timetables.
  + **Conflict Resolution**: Ensured there were no overlapping classes or scheduling conflicts.
* **UI Testing**:
  + Tested the Streamlit interface to ensure timetables were displayed correctly and that all admin functionalities (adding courses, modifying schedules) worked as expected.
* **Refinements**:
  + Refined the genetic algorithm by adjusting parameters (population size, mutation rate) to improve the efficiency and quality of the generated timetables.
  + Added error handling and user feedback to the UI for better user experience.

Future scope

### **Slide 11: Future Scope**

#### **Scalability:**

* The system can be extended to handle multiple campuses, additional rooms, or more complex scheduling needs like different types of courses (lectures, labs, etc.).

#### **Additional Features:**

* **Mobile App Integration**: Create an app for students to view their schedules and make requests for changes.
* **Enhanced Reporting**: Generate detailed reports on course scheduling, instructor workload, and room usage.

#### **Integrations:**

* Integration with campus management systems for real-time scheduling and updates.

Conclusion

### **Slide 12: Conclusion**

#### **Usages:**

* Educational institutions, such as universities and colleges, can benefit from automating the scheduling process, saving time and resources.

#### **Target Audience:**

* Schools, universities, and educational administrators are looking to streamline the timetable creation process.

#### **Self-made Project:**

* Developed from scratch, this project demonstrates skills in genetic algorithms, database management, and UI/UX design.