University Timetabling Description

UniTime is a comprehensive educational scheduling system that supports developing course and exam timetables, managing changes to these timetables, sharing rooms with other events, and scheduling students to individual classes. It is a distributed system that allows multiple university and departmental schedule managers to coordinate efforts to build and modify a schedule that meets their diverse organizational needs while allowing for minimization of student course conflicts. It can be used alone to create and maintain a school's schedule of classes and/or exams, or interfaced with an existing student information system.

It consists mainly of 4 components which are: Courses, Students, Exams and Events. Course timetabling and management is found within the Courses components as it is the main feature for managing courses on the program, Examination timetabling is found in the exams part of the project, Event management is found in events class and Student Scheduling is found in students. These represent the most basic and important features in the program along other features added by developers across the lifetime of the open source project.

While reviewing this project, we followed the Opportunistic Approach for code comprehension as it provides the positive side of both the bottom-up approach and the top-down one. We are not familiar with this kind of software as we haven’t worked on something similar before so reviewing the documents included first and gaining an overview of the functions of the program was a good way to start getting to know the project so we began with the top-down approach, then we selectively applied the bottom-up strategy when we reviewed the different classes in the code to verify the hypothesis resulted from our research in the top-down approach reading. Applying this approach ensured that we know what components are essential to exist and must be present and what these components consist of in the first place and use the hypothesis we deduced from reading to confirm our understanding of the program. So the presence of beacons was essential here as we are still not 100% sure of the strategy to be executed so having them would be a flexible approach to a no-current strategy situation such as the present by suggesting a hypothesis that is best verified by non-current strategy.