1 Formal Requirements Definition

1.1 Requirements

	Performance Levels	**F**	**NF**	**M**	**O**
1.	The system should be able to handle the traffic that is expected without severe drops in performance		X		X
2.	The system should be built and optimised according to the performance of the computing device architecture provided by the university in order to ensure that all students are able to easily access their timetables		X	X	
3.	The algorithm to create examination-venue-time slot combinations should run efficiently		X	X	

	Reliability	**F**	**NF**	**M**	**O**
1.	The system should provide reliable access to students by being constantly available (aside from necessary maintenance downtime)		X	X	
2.	The system should maintain data consistency so as to ensure that functional requirements are carried out with the most recent and correct data		X	X	

	Security/Privacy	**F**	**NF**	**M**	**O**
1.	A student should have to login (with a valid username and password) before private information is accessible		X	X	
2.	User levels can be used by the system to distinguish between students and administrators, offering more options to the administrator and being more restricted towards the student		X	X	
3.	User levels can be used by the system to distinguish between students and administrators, offering more options to the administrator and being more restricted towards the student		X	X	

	Quality	**F**	**NF**	**M**	**O**
1.	The system should maintain professional build quality which may encourage implementation and use of the platform		X	X	

1.2 Assumptions

The list below contains some constraints (or limitations) that affect the system:

- The system has access to the institutions personnel and student databases - students, course coordinators/lecturers and administrators do not have to register in order to use the system.

1.3 Dependencies

The list below contains some functionalities that the system depends on:

- A valid course listing/database with related courses (i.e. courses that can be done simultaneously) clearly indicated - the number of clashes which the system should minimise and generation of a timetable is dependent on this.
- Internet connection - the system is based on a client-server architecture with the server being centralised and remote.

- Google Calendar API - if the API drastically changes, the system won't be able to add exam events for students, lecturers and course coordinators.

1.4 Operating Environment

- **Architecture**: web-based application (client-server) - three-tier
- **Operating System**: Windows, Linux, MacOS, Android, iOS - any OS wherein a web browser can be used since a web application is being developed
- **Database**: SQLite database
- **Database Type**: centralised
- **Platform**: Python/Javascript/HTML/CSS - used to develop a web app to be accessed through a web browser

1.5 Data Management

Data management details how the input and output of the system will be stored. The majority of the data relating to the operation of the SchedEx system will be stored in a relational database. The details of the staff and students are stored in the database of the institution at which the system is being used. The data which is received and generated by the SchedEx system (e.g. course preferences and timetables) is also stored in a database and is received using the business layer of the three-tiered web application.

1.6 Portability

The system should be able to run on any platform with a web browser.