Board of Studies
Course Proposal Template

PROPOSED COURSE TITLE: Informatics Large Practical

PROPOSER(S): Stephen Gilmore

DATE: March 26, 2018

SECTION 1 – CASE FOR SUPPORT

[*This section should summarise why the new course is needed, how it fits with the existing course portfolio, the curricula of our Degree Programmes, and delivery of teaching for the different years it would affect.*]

1a. Overall contribution to teaching portfolio

[*Explain what motivates the course proposal, e.g. an emergent or maturing research area, a previous course having become outdated or inappropriate in other ways, novel research activity or newly acquired expertise in the School, offerings of our competitors.*]

The School of Informatics presently offers three large practicals for third-year undergraduate students. These are the Computer Science Large Practical (INFR09044), Software Engineering Large Practical (INFR09045), and AI Large Practical (INFR09043). This is a proposal for a single unified large practical to replace these three separate large practicals. The three large practicals have a lot of academic goals in common and the present system of having three separate practicals has a number of disadvantages:

(i) student enrolment cannot be automatic since the practicals are not compulsory;

(ii) students are sometimes enrolled on the wrong practical by their personal tutors;

(iii) students sometimes take a long time to decide which of the practicals to do, and change late in the semester;

(iv) there is perceived to be a difference in the level of difficulty between the large practicals, whereas they should really all be at the same level; and

(v) it seems to be more difficult for the students to choose between the three practicals because they are so similar.

Replacing the three practicals with a single compulsory one would address these problems and make student course administration simpler and more lightweight. There is a single compulsory group practical in second semester so having a single compulsory individual practical in the first semester can also be argued to make the degree programme more coherent and easier to understand.

1b. Target audience and expected demand

[*Describe the type of student the course would appeal to in terms of background, level of ability, and interests, and the expected class size for the course based on anticipated demand. A good justification would include some evidence, e.g. by referring to projects in an area, class sizes in similar courses, employer demand for the skills taught in the course, etc.*]

The course should appeal to all students who wish to develop their proficiency in software development and their skills in planning and carrying out an individual project which is a 50%-scale 20 points version of the 40 points Honours project (or MInf Part 1 project) which they will do in Year 4.

Because the course is compulsory for all Year 3 students we can estimate it to have an enrolment which is similar in size to the School of Informatics students enrolled for Informatics 2A or Informatics 2B in 2017–2018 (around 230 students) with a reduction for the students who will not pass the Year 2 exams, so we might estimate 200 Year 3 students taking the Informatics Large Practical in 2018–2019. The course lectures will need to be scheduled in a lecture theatre of this sort of capacity.

1c. Relation to existing curriculum

[*This section should describe how the proposed course relates to existing courses, programmes, years of study, and specialisms. Every new course should make an important contribution to the delivery of our Degree Programmes, which are described at* `http://www.drps.ed.ac.uk/17-18/dpt/drps_inf.htm`*. Please name the Programmes the course will contribute to, and justify its contribution in relation to courses already available within those programmes. For courses available to MSc students, describe which specialism(s) the course should be listed under (see* `http://web.inf.ed.ac.uk/infweb/student-services/ito/students/taught-msc-2017/programme-guide/specialist-areas`*), and what its significance for the specialism would be. Comment on the fit of the proposed course with the structure of academic years for which it should be offered. This is described in the Year Guides linked from* `http://web.inf.ed.ac.uk/infweb/student-services/ito/students`*.*]

The course fulfils the contribution to the curriculum which the separate Large Practicals make at present. The course tries to solidify the skills in programming which the students have developed in years 1 and 2 to improve their ability and confidence in tackling larger software development tasks which are typically found in courses such as the System Design Project in Semester 2 and the Honours Project in Year 4.

1d. Resources

[*While course approvals do not anticipate the School's decision that a course will actually be taught in any given year, it is important to describe what resources would be required if it were run. Please describe how much lecturing, tutoring, exam preparation and marking effort will be required in steady state, and any additional resources that will be required to set the course up for the first time. Please make sure that you provide estimates relative to class size if there are natural limits to its scalability (e.g. due to equipment or space requirements). Describe the profile of the course team, including lecturer, tutors, markers, and their required background. Where possible, identify a set of specific lecturers who have confirmed that they would either like to teach this course apart from the proposer, or who could teach the course in principle. It is useful to include ideas and suggestions for potential teaching duty re-allocation (e.g. through course sharing, discontinuation of an existing course, voluntary teaching over and above normal teaching duties) to be taken into account when resourcing decisions are made.*]

The course would require one lecturer. The course has no examination so the marking effort of the course comes from the two submissions from the students for Part 1 and Part 2 of the practical. Part 1 carries much less weight than Part 2, so we estimate that it can be marked more quickly. Estimating a class size of 200 students, we have this:

200 submissions for Part 1 at 30 minutes per submission:	100 hours marking
200 submissions for Part 2 at 60 minutes per submission:	200 hours marking
Total:	300 hours marking

In order to meet the expected return rate for coursework, this effort would be best distributed across a marking team thus:

10 markers	20 Part 1s + 20 Part 2s	30 hours per person contract	Return in 2 weeks
8 markers	25 Part 1s + 25 Part 2s	37.5 hours per person contract	Return in 3 weeks
5 markers	40 Part 1s + 40 Part 2s	60 hours per person contract	Return in 4 weeks
4 markers	50 Part 1s + 50 Part 2s	75 hours per person contract	Return in 5 weeks

With fewer than eight markers on the marking team it would be unlikely that the scripts would be returned in time.

The course should be supported by two hours of lab sessions a week (either two one-hour lab sessions, or one two-hour lab session). Labs should start in Week 3 of the semester and run for ten weeks. Two demonstrators would be required for this lab if available; otherwise one of these roles could be taken by the course lecturer and the other by a TA for the course.

Two demonstrators	Two hours per week	For 10 weeks	40 hours of demonstrating
One TA + the lecturer	Two hours per week	For 10 weeks	20 hours of TA time

Putting this all together, the preferred course personnel would likely be one lecturer, 10 markers, and one TA.

Suitable lecturers for this course would be those who already have experience in running an individual Large Practical.

Because this course is replacing three others, the lecturer for the course could be one of the Large Practical lecturers who are released from those duties. It is not necessary for this course to be taught on a voluntary basis over-and-above normal teaching duties.

Another resource implication to consider is that we can now factor in to each student's DICE quota allocation the space necessary to store the student's copy of the software needed for the course. The average installation of the software used on the course requires 5Gb, so it would be desirable to increase the DICE disk quota for all Year 3 students by 5Gb.

SECTION 2 – COURSE DESCRIPTOR

[*This is the official course descriptor that will be published by the University and serves as the authoritative source of information about the course for student via DRPS and PATH. Current course descriptions in the EUCLID Course Catalogue are available at www.euclid.ed.ac.uk under 'DPTs and Courses', searching for courses beginning 'INFR'.*]

2a. Course Title [Name of the course.]: Informatics Large Practical

2b. SCQF Credit Points: 20

[*The Scottish Credit and Qualifications Framework specifies where each training component provided by educational institutions fits into the national education system. Credit points per course are normally 10 or 20, and a student normally enrols for 60 credits per semester. For those familiar with the ECTS system, one ECTS credit is equivalent to 2 SCQF credits. See also* `http://scqf.org.uk/the-framework/scqf-credit-points/`.]

SCQF Credit Level: 9

[*These levels correspond to different levels of skills and outcomes, see* `http://www.sqa.org.uk/files_ccc/SCQF-LevelDescriptors.pdf` *At University level, Year 1/2 courses are normally level 8, Year 3 can be level 9 or 10, Year 4 10 or 11, and Year 5/MSc have to be level 11. MSc programmes may permit a small number (up to 30 credits overall) of level 9 or 10 courses.*]

Normal Year Taken: 3

[*While a course may be available for more than one year, this should specify when it is normally taken by a student. "5" here indicates the fifth year of undergraduate Masters programmes such as MInf.*]

Also available in years: None

[*Different options are possible depending on the choice of SCQF Credit Level above: for level 9, you should specify if the course is for 3rd year undergraduates only, or also open to MSc students (default); for level 10, you should specify if the course is available to 3rd year and 4th year undergraduates (default), 4th year undergraduates only, and whether it should be open to MSc students; for level 11, a course can be available to 4th and 5th year undergraduates and MSc students (default), to 5th year undergraduates and MSc students, or to MSc students only*]

Undergraduate or Postgraduate? Undergraduate

[*If the course is only available to MSc students, then it must be classified as a Postgraduate course. All other courses, regardless of level, will be classified as Undergraduate.*]

2c. Subject Area and Specialism Classification:

[*Any combination of Computer Science, Artificial Intelligence, Software Engineering and/or Cognitive Science as appropriate. For courses available to MSc students, please also specify the relevant MSc specialist area (to be found in the online MSc Year Guide at* `http://web.inf.ed.ac.uk/infweb/student-services/ito/students/taught-msc-2017/programme-guide/specialist-areas`*), distinguishing between whether the course should be considered as "core" or "optional" for the respective specialist area.*]

- Computer Science,
- Software Engineering,
- Artificial Intelligence, and
- Cognitive Science

Appropriate/Important for the Following Degree Programmes:

[*Please check against programmes from* `http://www.drps.ed.ac.uk/17-18/dpt/drps_inf.htm` *to determine any specific programmes for which the course would be relevant (in many cases, information about the Subject Area classification above will be sufficient and specific programmes do not have to be specified). Some courses may be specifically designed for non-Informatics students or with students with a specific profile as a potential audience, please describe this here if appropriate.*]

- Artificial Intelligence (BSc Hons)
- Artificial Intelligence and Computer Science (BSc Hons)
- Artificial Intelligence and Mathematics (BSc Hons)
- Artificial Intelligence and Software Engineering (BEng Hons)
- Artificial Intelligence with Management (BEng Hons)
- Cognitive Science (BSc Hons)
- Computer Science (BEng Hons)
- Computer Science (BSc Hons)
- Computer Science and Electronics (BEng Hons)
- Computer Science and Management Science (BSc Hons)
- Computer Science and Mathematics (BSc Hons)
- Computer Science and Physics (BSc Hons)
- Computer Science with Management (BEng Hons)
- Informatics (MInf)
- Software Engineering (BEng Hons)
- Software Engineering with Management (BEng Hons)

Timetabling Information:

[*Provide details on the semester the course should be offered in, specifying any timetabling constraints to be considered (e.g. overlap of popular combinations, other specialism courses, external courses etc).*]

The course should be offered in Semester 1 in order to help prepare students for the System Design Project in Semester 2.

The course should have two lectures a week. The lectures could be timetabled in the slots which are currently occupied by SELP and CSLP (Wed 12:10–13:00 and Fri 12:10–13:00).

2d. Summary Course Description:

[*Provide a brief official description of the course, around 100 words. This should be worded in a student-friendly way, it is the part of the descriptor a student is most likely to read.*]

The Informatics Large Practical exposes students to the problems that arise with the design and implementation of large-scale software systems, and to methods of coping with such problems. Students will gain experience in how to:

- Schedule their work load
- Design clearly and coherently structured systems
- Discover and use relevant technical information
- Implement a large-scale software system
- Design and run experiments and tests
- Analyse and report results
- Present their work in a clear and concise way.

Course Description:

[*Provide an academic description, an outline of the content covered by the course and a description of the learning experience students can expect to get. See guidance notes at:* `http://www.studentsystems.is.ed.ac.uk/staff/Support/User_Guides/CCAM/CCAM_Information_Captured.html`]

The Informatics Large Practical gives students experience in developing a non-trivial software system and reporting on the end product. In this way, the practical provides an introduction to the issues and requirements of the more demanding fourth-year project. In particular, the student gains practical experience of:

- Reading technical material and identifying the important content
- Identifying and formalising project requirements
- Identifying computational problems and inventing algorithmic solutions

- Constructing a detailed design which does not over-commit to implementation detail
- Implementing and testing a software application which realises the design
- Experimenting with the implementation to explore the solutions to the computational problems
- Writing a report which documents the solutions and the implementation
- Managing a software project using a source-code repository.

Pre-requisite Courses:

[*Specify any courses that a student must have taken to be permitted to take this course. Pre-requisites listed in this section can only be waived by special permission from the School's Curriculum Approval Officer, so they should be treated as "must-have". By default, you may assume that any student who will register for the course has taken those courses compulsory for the degree for which the course is listed in previous years. Please include the FULL course name and course code.*]

None.

Co-requisite Courses:

[*Specify any courses that should be taken in parallel with the existing course. Note that this leads to a timetabling constraint that should be mentioned elsewhere in the proposal. Please include the FULL course name and course code.*]

None

Prohibited Combinations:

[*Specify any courses that should not be taken in combination with the proposed course. Please include the FULL course name and course code.*]

Students MUST NOT also be taking Informatics Research Review (INFR11136) OR Informatics Project Proposal (INFR11147).

Students MUST NOT previously have passed AI Large Practical (INFR09043) OR Computer Science Large Practical (INFR09044) OR Software Engineering Large Practical (INFR09045).

Other Requirements:

[*Please list any further background students should have, including, for example, mathematical skills, programming ability, experimentation/lab experience, etc. It is important to consider that unless there are formal prerequisites for participation in a course, other Schools can register their students onto our courses, so it is important to be clear in this section. Also be aware that MSc students are unlikely to have the pre-requisite courses, so alternative knowledge should be recommended. If you want to only*

permit this by special permission, a statement like "Successful completion of Year X of an Informatics Single or Combined Honours Degree, or equivalent by permission of the School." can be included.]

Successful completion of Year 2 of an Informatics Single or Combined Honours Degree, or equivalent by permission of the School. For external students where this course is not listed in your DPT, please seek special permission from the course organiser.

Available to Visiting Students: No

[*Provide a justification if the answer is No.*]

This course is not available to visiting students because it is resource-intensive to deliver and will already have a sufficiently large cohort of students taking the course because it is compulsory.

2e. Summary of Intended Learning Outcomes (MAXIMUM OF 5):

[*List the learning outcomes of the course, emphasising what the impact of the course will be on an individual who successfully completes it, rather than the activity that will lead to this outcome. Further guidance is available from* `https://canvas.instructure.com/courses/801386/files/24062695`]

On completion of the Informatics Large Practical, the student will be able to:

1. Consider alternative algorithm designs and data structures for tackling a given problem.
2. Show awareness of the difference between design and implementation in software development.
3. Implement and debug a software system of medium to large size.
4. Design and carry out experiments and tests, and explain the methodology involved.
5. Write a well-structured report providing clear and concise documentation for a software project.

Assessment Information

[*Provide a description of all types of assessment that will be used in the course (e.g. written exam, oral presentation, essay, programming practical, etc) and how each of them will assess the intended learning outcomes listed above. Where coursework involves group work, it is important to remember that every student has to be assessed individually for their contribution to any jointly produced piece of work. Please include any minimum requirements for assessment components e.g. student must pass all individual pieces of assessment as well as course overall.*]

One large design, implementation and evaluation project, done in two parts.

1. The first part consists of a project plan outlining the problem area, and proposing a solution technique and application design considering both functional and non-functional requirements on the project (25% of course total).

2. In the second part, students fully implement their application design, and submit both their implementation and a report that presents and analyses their specification, design, implementation and tests (75% of course total).

Assessment Weightings:

Written Examination:	0%
Practical Examination:	0%
Coursework:	100%

Time spent on assignments:

[*Weightings up to a 70/30 split between exam and coursework are considered standard, any higher coursework percentage requires a specific justification. The general expectation is that a 10-point course will have an 80/20 split and include the equivalent of one 20-hour coursework assignment (although this can be split into several smaller pieces of coursework. The Practical Examination category should be used for courses with programming exams. You should not expect that during term time a student will have more than 2-4 hours to spend on a single assignment for a course per week. Please note that it is possible, and in many cases desirable, to include formative assignments which are not formally assessed but submitted for feedback, often in combination with peer assessment.*]

This course is assessed by a practical assignment only; there is no examination. As a 20-point course, this should occupy approximately 200 hours of effort from the students. The course has eight hours of lectures and twenty hours of labs available for each student, leaving 172 hours available for practical work.

Of this remaining time, 10 hours are devoted to self-directed study which is used for reading papers or technical documentation as necessary to prepare for the practical work of the project. This time also includes the time needed to install the software needed to do the implementation work of the practical.

After this, we envision 40 hours of work for Part 1 and 120 hours of work for Part 2 (this effort is split in accordance with the 25%:75% weighting of the two parts).

Part 1 is done over a three-week period. Part 2 is done over a nine-week period. (This time in the semester is also split in accordance with the 25%:75% weighting of the two parts).

Two hours are for feedback/feedforward from Part 1 and Part 2.

Academic description:

[*A more technical summary of the course aims and contents. May include terminology and technical content that might be more relevant to colleagues and administrators than to students.*]

This is a practical course which encourages independent learning and creativity through the ability to shape the coursework exercise within defined limits. Students are encouraged to add bonus features to their solutions to make them different from those of others and to do independent research into how to realise these bonus features. The course thus contains a (limited) amount

of adaptation of the kind which is typically found in an Honours project where the student may alter the course of the project as new goals appear.

Syllabus:

[*Provide a more detailed description of the contents of the course, e.g. a list of bullet points roughly corresponding to the topics covered in each individual lecture/tutorial/coursework. The description should* ***not exceed 500 words*** *but should be detailed enough to allow a student to have a good idea of what material will be covered in the course. Please keep in mind that this needs to be flexible enough to allow for minor changes from year to year without requiring new course approval each time.*]

1. Introduction to the problem domain and the coursework specification. (1 lecture)
2. Introduction to the software used on the course. (1 lecture)
3. Project planning and management including source code control. (1 lecture)
4. Introduction to software testing, and building dependable systems. (1 lecture)
5. Problem-domain specific material as appropriate to the practical. (3 lectures)
6. Documentation and report writing. (1 lecture)

Relevant QAA Computing Curriculum Sections:

[*Please see* `http://www.qaa.ac.uk/en/Publications/Documents/SBS-Computing-consultation-15.pdf` *to check which section the course fits into.*]

Artificial Intelligence, Computer Based Systems, Software Engineering, Systems Analysis and Design, Professionalism.

Graduate Attributes, Personal and Professional skills:

[*This field should be used to describe the contribution made to the development of a student's personal and professional attributes and skills as a result of studying this course – i.e. the generic and transferable skills beyond the subject of study itself. Reference in particular should be made to SCQF learning characteristics at the correct level* `http://www.sqa.org.uk/files_ccc/SCQF-LevelDescriptors.pdf`.]

This course attempts to enhance the student's attributes, personal and professional skills in the following areas. Students on the course will:

- Apply knowledge, skills and understanding:
  - In using a range of the principal professional skills, techniques, and practices associated with Informatics. (For example, software installation, software development skills, testing strategies and frameworks.)
  - In using a few techniques and practices that are specialised. (For example, specific testing strategies for applications with rich user interface elements.)

  - In practising routine methods of enquiry and research. (For example, through using search engines to diagnose problems with software configuration or installation, or using question-and-answer fora to solve implementation issues.)

- Undertake critical analysis, evaluation and/or synthesis of ideas, concepts, information and issues in Informatics. (For example, through researching algorithms which could be used in the service of this practical.)
- Identify and analyse routine professional problems and issues. (For example, through gaining experience in understanding problems with complex software systems, and in improving debugging skills in learning how to use sources of information such as StackOverflow in debugging programs.)
- Draw on a range of sources in making judgements. (For example, reading documentation on software, learning about software frameworks or libraries in trying to decide whether or not to use them in this practical exercise.)
- Use a wide range of routine skills and some advanced and specialised skills in support of established practices in Informatics, for example:
  - Present or convey, formally and informally, information on standard/mainstream topics in Informatics to a range of audiences.
  - Use a range of software applications to support and enhance work. (For example, integrated development environments, device managers, SDK managers, emulators, physical devices, debuggers, profilers, testing frameworks.)
  - Interpret, use and evaluate numerical and graphical data to achieve goals/targets. (For example, using profilers to identify bottlenecks in software.)
- Exercise autonomy and initiative in some activities at a professional level in Informatics. (For example, in deciding how to extend the project with bonus features which enhance the value of the project.)

Reading List:

[*Provide a list of relevant readings. See also remarks under 3d.*]

None.

Breakdown of Learning and Teaching Activities:

[*Total number of lecture hours and tutorial hours, with hours for coursework assignments. The breakdown of learning and teaching activities should only include contact hours with the students; everything else should be accounted for in the Directed Learning and Independent Learning hours. The total being 10 × course credits. Assume 10 weeks of lectures slots and 10 weeks of tutorials, though not all of these need to be filled with actual contact hours. As a guideline, if a 10-pt course has 20 lecture slots in principle, around 15 of these should be filled with examinable material; the rest should be used for guest lectures, revision sessions, introductions to assignments, etc. Additional categories of learning and teaching activities are available, a full list can be found at:* `http://www.euclid.ed.ac.uk/Staff/`

Support/User_Guides/CCAM/Teaching_Learning.htm. You may also find the guidance on "Total Contact Teaching Hours" and "Examination & Assessment Information" at: http://www.studentsystems.ed.ac.uk/Staff/Support/User_Guides/CCAM/CCAM_Information_Captured.html]

Lecture Hours:	8 hours
Seminar/Tutorial Hours:	0 hours
Supervised practical/Workshop/Studio hours:	20 hours
Summative assessment hours:	160 hours
Feedback/Feedforward hours:	2 hours
Directed Learning and Independent Learning hours:	10 hours
Total hours:	200 hours

Keywords:

[*A list of searchable keywords.*]

Practical exercise; software development; specification, design, and implementation; software testing; documentation and reporting.

SECTION 3 – COURSE MATERIALS

3a. Sample exam question(s)

[*Sample exam questions with model answers to the individual questions are required for new courses. A justification of the exam format should be provided where the suggested format is non-standard. The online list of past exam papers gives an idea of what exam formats are most commonly used and which alternative formats have been* `http://www.inf.ed.ac.uk/teaching/exam_papers/`.]

There is no examination for this course.

3b. Sample coursework specification

[*Provide a description of a possible assignment with an estimate of effort against each sub-task and a description of marking criteria.*]

Large practical specifications typically run to several pages in length. Below is a short summary of a full specifiication.

> *For this practical exercise you are to create an Android mobile phone app for a multiplayer game which implements a location-based search in which players explore a map collecting virtual coins which can be stored in an online bank. In addition to collecting coins by walking to the coin's location on the map, the player also controls an airborne drone, which can collect coins at its location, which can be a limited distance from the player's own location. The virtual coins come from a set of (fictional) cryptocurrencies with relative values which fluctuate over time relative to a cryptocurrency standard. In order to pay the coins into the bank, a player must upload a proof-of-work which includes the path walked by the player, the flight path of the drone, and the coins which can be collected on the way. All three of the elements of the proof-of-work are subject to verification by the bank, and proof-of-work which fails this verification will be rejected with nothing added to the player's bank account. New maps are released each day and are valid only for a single day with their coins becoming worthless after that. The aim of the game is to become the player with the highest value virtual currency in the bank at any point (including the end date of the gameplay at the submission date of the practical). It is up to you to decide how the player controls the drone: whether it must be explicitly controlled by the player, with a remote-control style interface, giving instructions at each step; or has some degree of autonomy up to full autonomous flight with a homing function. You can also add other bonus features to your app to distinguish it from others.*
>
> ***Part 1 (25%):** Submit an interim report of up to five pages with a project plan and timeline.*
>
> ***Part 2 (75%):** Submit a final report of up to twenty-five pages together with your finished app.*

The coursework is designed to resemble a half-size Honours project which has a ten-page interim report and a fifty-page final report.

3c. Sample tutorial/lab sheet questions

[*Provide a list of tutorial questions and answers and/or samples of lab sheets.*]

There are no tutorials for this course. A sample lab sheet could contain details of installing the Android Studio software used for the course, getting a Google Maps Android API key, creating a GitHub repository to store the project source code, or running instrumented tests on the Android emulator.

3d. Any other relevant materials

[*Include anything else that is relevant, possibly in the form of links. If you do not want to specify a set of concrete readings for the official course descriptor, please list examples here.*]

- `https://developer.android.com/index.html`
- `https://developer.android.com/studio/index.html`
- `https://firebase.google.com`

SECTION 4 – COURSE MANAGEMENT

4a. Course information and publicity

[*Describe what information will be provided at the start of the academic year in which format, how and where the course will be advertised, what materials will be made available online and when they will be finalised. Please note that University and School policies require that all course information is available at the start of the academic year including all teaching materials and lecture slides.*]

A detailed description of the coursework specification will be provided at the start of the year in PDF document format. This document will have a version number and a change history as an appendix, allowing it to be updated if needed if errors are discovered in the specification, the details for submission with the `submit` command, or for other reasons. In order that all students see this document at the same time, this will be released on the day of the first lecture of the course.

Lecture slides in PDF format, one slide deck for each lecture, will be made available at the start of the academic year. These slides will be updated as necessary during the year if errors are found in the slides or if additional content is added to the slides.

Test data for the app will be released at the start of the year.

4b. Feedback

[*Provide details on feedback arrangements for the course. This includes when and how course feedback is solicited from the class and responded to, what feedback will be provided on assessment (coursework and exams) within what timeframe, and what opportunities students will be given to respond to feedback. The University is committed to a baseline of principles regarding feedback that we have to implement at every level. Further guidance is available from* `http://www.enhancingfeedback.ed.ac.uk/staff.html`.]

A standard mid-semester course survey will be distributed during the course lectures. Feedback from this will be distributed as a report which is made available from the course web page.

Feedback and the assessment result from the student's Part 1 submission will be made available as a written individual report sent by email. This will also contain feedforward suggestions relating to the Part 2 submission.

Feedback and the assessment result from the student's Part 2 submission will be made available as a written individual report sent by email.

Students can respond to the feedback via email to the course lecturer.

4c. Management of teaching delivery

[*Provide details on responsibilities of each course staff member, how the lecturer will recruit, train, and supervise other course staff, what forms of communication with the class will be used, how required*

equipment will be procured and maintained. Include information about what support will be required for this from other parties, e.g. colleagues or the Informatics Teaching Organisation.]

The course lecturer will ask the Informatics Teaching Organisation to help with recruiting a TA and markers for the course. The course lecturer will work with the TA to plan laboratory sessions and will meet with the markers to explain their duties. The ITO should divide the class into equally-sized tutorial groups, one for each marker. Coursework will be submitted electronically via the DICE `submit` command, which will file the submissions into the appropriate groups. The tutorial groups are notional in this case, because the course does not have tutorials, and are just a way of distributing the work between the markers. Thus, there will not be the usual overhead of students wishing to swap tutorial groups, since there is no tutorial group meeting time to clash with other timetabled commitments.

Markers will mark the submitted work and the results are to be entered via a webmark form based on the UG4/MInf marking form, with suitable modifications. These marks are moderated by the course lecturer before being released to the students.

The course will have a Piazza forum for discussion of software problems and coursework-related issues. Because not all students check Piazza regularly, official course announcements (administrative or otherwise) should go out via an `ilp-students@inf.ed.ac.uk` email list to ensure that they are received by all students. Marking-related emails should be sent out to an `ilp-tutors@inf.ed.ac.uk` emal list to ensure that they are received by all tutors.

There is no requirement to purchase and maintain special-purpose equipment for this practical.

SECTION 5 – COMMENTS

[*This section summarises comments received from relevant individuals prior to proposing the course. If you have not discussed this proposal with others please note this.*]

5a. Year Organiser Comments

[*Year Organisers are responsible for maintaining the official Year Guides for every year of study, which, among other things, provide guidance on available course choices and specialist areas. The Year Organisers of all years for which the course will be offered should be consulted on the appropriateness and relevance on the course. Issues to consider here include balance of course offerings across semesters, subject areas, and credit levels, timetabling implications, fit into the administrative structures used in delivering that year.*]

5b. BoS Academic Secretary

[*Any proposal has to be checked by the Secretary of the Board of Studies prior to discussion at the actual Board meeting. This is a placeholder for their comments, mainly on the formal quality of the content provided above.*]