Class Syllabus

Instructors:
Dr. Mark Palmeri, M.D., Ph.D.
(mark.palmeri@duke.edu)
Office Hours: `https://calendly.com/mark-palmeri`

Suyash Kumar, Uber Software Engineer, Web Guru, Deep Learning Master
(suyash.kumar@duke.edu)
Office Hours: TBD (TBD)

Teaching Assistants:
Arjun Desai
(arjun.desai@duke.edu)
Office Hours: TBD (TBD)

Nisarg Shah
(nisarg.shah@duke.edu)
Office Hours: TBD (TBD)

Lecture: Wed/Fri 11:45–13:00, 216 Hudson Hall

Course Overview

Software plays a critical role in almost all medical devices, spanning device control, feedback and algorithmic processing. This course focuses on software design skills that are ubiquitous in the medical device industry, including software version control, unit testing, fault tolerance, continuous integration testing and documentation. Experience will be gained in Python and JavaScript (and potentially other languages).

The course will be structured around a project to build an Internet-connected medical device that measures and processes a biosignal, sends it to a web server, and makes those data accessible to a web client / mobile application. This project will be broken into several smaller projects to develop software design fundamentals. All project-related work will be done in groups of 3 students.

Prerequisites: Basic familiarity with programming concepts (e.g., variables, loops, conditional statements)

Course Objectives

- Define software specifications and constraints
- Device programming fundamentals
  - Review of data types
  - Python (v3.6): numpy, scipy, pandas, scikit
  - Virtual environments & depedency management (`pip`, `requirements.txt`)
  - Data management (variables, references, pointers, ASCII/Unicode/binary data)
  - Regular expressions (regex)
  - Use of a programming IDE
  - Debugging (`pudb`)
- Backend Software Development in the Cloud
  - Databases (MySQL, MongoDB)
  - HTTP & RESTful APIs
  - Leverage scalable compute infrastructure in the cloud via Remote Procedure Calls (RPCs)
  - Call web services from Matlab & Python
  - Design & Implementation of a biomedical web service (Python Flask)
  - Docker and dependency management intro
  - SSL and Encryption

  - Internet of Things (IoT) and cloud connected biomedical device design
  - Sockets and streaming data over networks
- Software version control (`git`, GitHub)
- Documentation
  - Docstrings
  - Markdown
  - Sphinx
  - `https://readthedocs.org`
- Testing
  - Unit testing
  - Functional / System testing
  - Continuous integration (Travis CI)
- Fault tolerance (raising exceptions)
- Logging
- Resource profiling (`cProfile`)

Attendance

Lecture attendance and participation is important because you will be working in small groups most of the semester. Participation in these in-class activities will count for 15% of your class grade. It is very understandable that students will have to miss class for job interviews, personal reasons, illness, etc. Absences will be considered *excused* if they are communicated to Dr. Palmeri and Mr. Kumar at least 48 hours in advance (subject to instructor discretion as an excused absence) or, for illness, through submission of Short Term Illness Form (STIF) **before** class. Unexcused absences will count against the participation component of your class grade.

Textbooks & References

There are no required textbooks for this class. A variety of online resources will be referenced throughout the semester. A great resource for an overview of Python:

`https://github.com/jakevdp/WhirlwindTourOfPython`

Project Details

Project details will be discussed in lecture throughout the semester.

Grading

The following grading scheme is subject to change as the semester progresses:

Participation	15%
Midterm project deliverables	55%
Final project	30%

Class Schedule

The course schedule is very likely to change depending on progress throughout the semester. Specific lecture details, along with deliverable due dates, will be updated on Sakai and this syllabus (`https://github.com/mlp6/Medical-Software-Design`). New due dates will be announced in lecture and by Sakai announcements that will be emailed to the class. The following is a summary of activities this semester:

Date	Lecture	Assignment
Fri Jan 12	Class Introduction, Objectives and Logistics [MLP]	A01: Git(Hub), Python Install
Wed Jan 17 Fri Jan 19	Git: Repo Setup, Issues, Branching, Pushing/Pulling [MLP] Python syntax, virtualenv (`pip`), Unit Testing [MLP]	 A02: Python Fundamentals
Wed Jan 24 Fri Jan 26	Travis CI, Python Exception Handling & Logging [MLP] Python: Dictionaries, Numpy Arrays, Binary Data; Docstrings & Sphinx [MLP]	 A03: Py.test
Wed Jan 31 Fri Feb 02	Classes: Encapsulation & Composition [MLP] Data Types, APIs & JSON [MLP]	 A04: Documentation
Wed Feb 07 Fri Feb 09	Python: Read/Writing Data (CSV, JSON, HDF5, MATv5) [MLP] TBD	 A05: Heart Rate Monitor
Wed Feb 14 Fri Feb 16	TBD TBD	TBD TBD
Wed Feb 21 Fri Feb 23	TBD TBD	TBD TBD
Wed Feb 28 Fri Mar 02	Intro to Web Services & cloud-connected devices [SK] Python Flask, API design, virtual machines (AWS, GCP) [SK]	 A06: Basic Flask Web Service
Wed Mar 07 Fri Mar 09	Microservices, Docker, Flask cont'd [SK] Introduction to Databases [SK]	 A07: Docker Server
Wed Mar 14 Fri Mar 16	SPRING BREAK SPRING BREAK	SPRING BREAK SPRING BREAK
Wed Mar 21 Fri Mar 23	Web, Mobile, Desktop clients (ReactJS) [SK] Web client (ReactJS) cont'd, final project intro [SK]	A08: Cloud Heart Rate Monitor
Wed Mar 28 Fri Mar 30	Software Documentation, Streaming Data, Cloud-connected Hardware, SSL [SK] IEC 62304 [MLP]	
Wed Apr 04 Fri Apr 06	Debugging TBD, final project work	Working Project Code
Wed Apr 11 Fri Apr 13	Project "Lab" Final project work	Refactor Project Code
Wed Apr 18 Fri Apr 20	Final project work Final project work	
Wed Apr 25	Final project due	

Distributed Version Control Software (git)

Software management is a ubiquitous tool in any engineering project, and this task becomes increasingly difficult during group development. Version control software has many benefits and uses in software development, including preservation of versions during the development process, the ability for multiple contributors and reviewers on a project, the ability to tag "releases" of code, and the ability to branch code into different functional branches. We will be using GitHub (`https://github.com`) to centrally host our git repositories.

- *All* software additions, modifications, bug-fixes, etc. need to be done in your repository.
- The "Issues" feature of your repository should be used as a "to do" list of software-related items, including feature enhancements, and bugs that are discovered.
- There are several repository management models that we will review in class, including branch-development models that need to be used throughout the semester.
- Instructors and teaching assistants will only review code that is committed to your repository (no emailed code!).
- All of the commits associated with your repository are logged with your name and a timestamp, and these cannot be modified. Use descriptive commit messages so that your group members, instructors, and teaching assistants can figure out what you have done!! You should not need to email group members when you have performed a commit; your commit message(s) should speak for themselves.
- Code milestones should be properly tagged.
- Write software testing routines early in the development process so that anyone in your group or an outsider reviewing your code can be convinced that it is working as intended.
- Modular, modular, modular.
- Document!
- Make commits small and logical; do them often!

We will review working with git repositories in lecture, and feedback on your software repository will be provided on a regular basis.

Duke Community Standard & Academic Honor

Engineering is inherently a collaborative field, and in this class, you are encouraged to work collaboratively on your projects. The work that you submit must be the product of your and your group's effort and understanding. All resources developed by another person or company, and used in your project, must be properly recognized.

All students are expected to adhere to all principles of the Duke Community Standard. Violations of the Duke Community Standard will be referred immediately to the Office of Student Conduct. Please do not hesitate to talk with your instructors about any situations involving academic honor, especially if it is ambiguous what should be done.