FAKULTAS ILMU KOMPUTER
UNIVERSITAS INDONESIA

Tugas Akhir
CSIM603026 – Arsitektur dan Pemrograman Aplikasi Perusahaan
Semester Ganjil 2017/2018
Deadline: Kamis, 7 Desember 2017

Sistem Informasi Akademik Future Generation (SIAK-FG)

Version: 1.1, 11 November 2017

Revisi:
- **Versi 1.0**
- **Versi 1.1**
  - Menambahkan penjelasan pada pendahuluan
  - Melengkapi penjelasan Aplikasi Universitas
  - Klarifikasi penjelasan Aplikasi Sekretariat
  - Melengkapi penjelasan bagian File Pendukung
  - Menambahkan penjelasan Deliverables

Pendahuluan

Indonesia memiliki lebih dari 4.500 perguruan tinggi. Angka tersebut terdiri dari perguruan tinggi negeri, perguruan tinggi swasta, dan perguruan tinggi agama atau perguruan tinggi di bawah kementerian. Masing-masing perguruan tinggi tersebut memiliki sistem akademik, baik *online* maupun *offline*, untuk mengurus urusan akademis mereka.

Menteri Riset Teknologi dan Perguruan Tinggi (Menristek) saat ini memiliki sebuah mimpi untuk untuk mengintegrasikan berbagai sistem informasi yang dimiliki oleh masing-masing perguruan tinggi. Anda bersama tim Anda ditugaskan untuk membantu Menristek mewujudkan mimpi tersebut. Integrasi dari seluruh sistem informasi tersebut disebut sebagai Sistem Informasi Akademik *Future Generation* (SIAK-FG).

Sebuah perguruan tinggi tentunya memiliki berbagai macam *business process* didalamnya. SIAK-FG didesain untuk dapat menunjang *business process* yang terjadi dalam seluruh perguruan tinggi yang terdapat di Indonesia. Oleh karena itu, diperlukan sistem yang memiliki fitur yang sesuai dengan standar seluruh perguruan tinggi di Indonesia. Fitur tersebut tertuang dalam aplikasi-aplikasi yang akan terdapat dalam SIAK-FG. Daftar aplikasi tersebut adalah sebagai berikut:

- **Aplikasi Kurikulum**, digunakan oleh Kepala Program Studi

- **Aplikasi Penilaian**, digunakan oleh Dosen
- **Aplikasi Akademik Mahasiswa**, digunakan oleh Mahasiswa dan Sekretariat Akademik[Revisi 1.1]
- **Aplikasi Sekretariat**, digunakan oleh Sekretariat Akademik

- **Aplikasi Universitas**, digunakan oleh Staf Universitas

Kelima aplikasi tersebut akan di-*deploy* ke lima mesin yang berbeda. Aplikasi memiliki *user interface* untuk masing-masing pengguna dan memiliki *database* masing-masing yang saling terpisah. Oleh karena itu, setiap aplikasi saling *silo* dan perlu berinteraksi menggunakan *web service* untuk menjalankan fitur masing-masing.

Anda beserta seluruh anggota tim besar Anda akan membantu membuat seluruh aplikasi, kecuali Aplikasi Universitas, yang dibutuhkan SIAK-FG. Untuk setiap tim kecil dalam tim besar, akan bertanggung jawab dalam implementasi aplikasi yang terdapat dalam SIAK-FG. Perlu diperhatikan bahwa setiap aplikasi harus dapat saling terintegrasi, sehingga perlu adanya koordinasi antar tim kecil dalam sebuah tim besar dalam mengerjakan proyek SIAK-FG.

Basis Data

Anda dan tim tidak diberikan penjelasan tentang basis data yang akan digunakan. Oleh karena itu, Anda perlu mendefinisikan basis data Anda sendiri dari setiap proses bisnis yang diberikan.

Anda dipersilakan untuk membuat rancangan *database* Anda sendiri sesuai dengan analisis fitur yang Anda lakukan. Sebagai petunjuk, Anda dapat mengobservasi bagaimana proses bisnis SIAK-NG saat ini berjalan.

Deskripsi Kebutuhan Fitur Setiap Aplikasi

Aplikasi Universitas (dikembangkan oleh Tim Asisten)
Aplikasi Universitas akan dikembangkan oleh Tim Asisten untuk melayani data Universitas, Fakultas, Program Studi, dan ~~Mahasiswa~~[Revisi 1.1].

API Base URL: **https://apap2017-univ-apps.herokuapp.com/** [Revisi 1.1]

Aplikasi Universitas memiliki API:
- GET **getUniversitasList**
- GET **getUniversitas**

- GET **getFakultasList**
- GET **getFakultas**
- GET **getProdiList**
- GET **getProdi**

Dokumentasi API Aplikasi Universitas**:**
https://github.com/apap-2017/univ-app [Revisi 1.1]

Aplikasi Kurikulum

Aplikasi Kurikulum digunakan untuk mengelola kurikulum dan mata kuliah yang ada pada sebuah program studi. Aplikasi ini digunakan oleh Kaprodi untuk mengelola kurikulum dan mata kuliah sebuah program studi di suatu fakultas di sebuah universitas.

Sebuah kurikulum merupakan kumpulan mata kuliah yang dikelompokkan dalam *term*, misal *term* 1 sampai *term* 8. Kumpulan mata kuliah tersebut harus didefinisikan terlebih dahulu sebelum dapat disertakan dalam kurikulum. Kurikulum memiliki kaitan erat dengan program studi (prodi). Setiap prodi pasti memiliki paling tidak satu kurikulum yang diterapkan. Selain itu, satu prodi mungkin saja memiliki lebih dari satu kurikulum yang diterapkan untuk angkatan yang berbeda. Sebuah kurikulum memiliki informasi seperti jumlah SKS total, misalkan 144 SKS, tahun kurikulum, kumpulan mata kuliah pada kurikulum, dll. Salah satu fungsi penting lainnya dari kurikulum adalah menentukan kelulusan, wisuda, mahasiswa. Mahasiswa yang telah mencapai SKS untuk batas lulus dan telah lulus mata kuliah wajib dan pilihan tertentu statusnya lulus.

Sebuah mata kuliah memiliki informasi seperti jumlah SKS, prasyarat, nama, masuk pada salah satu kurikulum, dll. Sebuah mata kuliah mungkin memiliki prasyarat mata kuliah lain. Sehingga untuk mengambil mata kuliah tersebut mahasiswa perlu memenuhi kelulusan seluruh mata kuliah prasyarat. Mata kuliah dibagi menjadi mata kuliah wajib dan mata kuliah pilihan.

Aplikasi Kurikulum memiliki kapabilitas untuk:

- **Mengelola kurikulum, CRUD**
  Kurikulum berisi nama kurikulum, jumlah SKS wajib, jumlah SKS pilihan, mata kuliah wajib, mata kuliah pilihan, dan data lainnya jika diperlukan.

- **Mengelola mata kuliah dan prasyaratnya, CRUD**
  Mata kuliah memiliki nama, jumlah SKS, deskripsi, daftar prasyarat mata kuliah yang perlu diambil sebelumnya, dan data lainnya jika diperlukan.

- **Menampilkan semua angkatan aktif dan kurikulum yang digunakan angkatan tersebut dari suatu prodi (butuh API Aplikasi Universitas, API Aplikasi Kurikulum, dan API Aplikasi Akademik Mahasiswa)**
  Menampilkan daftar angkatan dan kurikulum yang digunakan oleh angkatan tersebut pada suatu prodi.

Aplikasi Kurikulum hanya menyimpan data:
- Kurikulum
- Mata Kuliah

Aplikasi Kurikulum dapat memberikan data kurikulum, mata kuliah, dan informasi prasyarat mata kuliah ke Aplikasi lain melalui **REST API *service***.

Aplikasi Kurikulum membutuhkan data universitas, fakultas, dan prodi dari **Aplikasi Universitas** dalam pembuatan kurikulum untuk membedakan kepemilikan kurikulum karena aplikasi ini akan diakses oleh ribuan universitas. Aplikasi Kurikulum juga membutuhkan data angkatan dan kurikulum yang digunakan angkatan tersebut dari **Aplikasi Sekretariat**. Selain dua kebutuhan diatas, **Aplikasi Kurikulum mungkin saja membutuhkan data dari aplikasi lainnya.**

Beberapa **contoh API yang mungkin dimiliki** oleh Aplikasi Kurikulum dan dapat diakses aplikasi lain:
- **/api/getKurikulum**
  Mengembalikan data kurikulum termasuk seluruh mata kuliah di kurikulum tersebut

- **/api/getMataKuliah**
  Mengembalikan data sebuah mata kuliah

- **/api/getPrasyarat**
  Mengembalikan prasyarat dari suatu mata kuliah

Anda dapat menyesuaikan API yang akan dibuat sesuai dengan kebutuhan dan kesepakatan dengan kelompok lain.

Aplikasi Sekretariat
Aplikasi Sekretariat digunakan membuat *term* atau semester perkuliahan, ganjil atau genap, dan membuat jadwal kelas perkuliahan. Aplikasi Sekretariat digunakan oleh staf akademik suatu fakultas.

Term merupakan rentang waktu masa perkuliahan, misalnya perkuliahan term ganjil dari tanggal 1 September - 24 Desember. *Term* juga memiliki informasi masa pengisian IRS mahasiswa. Masa pengisian IRS hanya satu hari saja dalam rentang waktu *term* misalkan hanya tanggal 1 September saja. Setelah itu IRS tidak dapat diubah lagi oleh mahasiswa.

Pembuatan jadwal kelas oleh staf akademik berpatokan dari kurikulum yang diadopsi oleh prodi tersebut dan term yang berlaku. Sehingga sebuah kelas dari mata kuliah genap tidak bisa diselenggarakan pada term ganjil, juga sebaliknya. Setiap kelas yang dibuat memiliki informasi term, nama kelas, waktu kelas, dll.

Staf sekretariat juga dapat melakukan pengaturan kurikulum yang dipakai untuk setiap angkatan di suatu prodi. Jadwal kelas yang dapat dibuat tergantung pengaturan dari kurikulum setiap angkatan. Sehingga jadwal kelas yang ditawarkan untuk setiap angkatan mungkin berbeda karena perbedaan kurikulum yang digunakan.

Aplikasi Sekretariat memiliki kapabilitas untuk:

- **Mengelola term, CRUD**
  Term memiliki informasi nomor term, range waktu term dilaksanakan, jadwal pengisian IRS, term ganjil atau genap, dan data lainnya jika diperlukan.

- **Mengelola jadwal kelas, CRUD (butuh API Aplikasi Kurikulum)**
  Kelas memiliki informasi nama kelas, mata kuliah, daftar jadwal kelas, dosen pengajar, dan data lainnya jika diperlukan.

- **Mengelola pengaturan kurikulum yang digunakan oleh masing-masing angkatan dari suatu prodi (butuh API Aplikasi Kurikulum, API Aplikasi Akademik Mahasiswa, dan API Universitas** [Revisi 1.1]**)**
  Untuk mengaplikasikan kurikulum kepada suatu angkatan perlu dilakukan *assignment* oleh staf sekretariat. Kurikulum diaplikasikan ke setiap mahasiswa pada angkatan tersebut, sehingga ketika mahasiswa tersebut mengisi IRS pilihan mata kuliah sudah sesuai dengan kurikulum yang diaplikasikan ke mahasiswa tersebut.

Aplikasi Sekretariat hanya menyimpan data:

- Term
- Kelas
- Pengaturan Kurikulum setiap angkatan dan individu

Aplikasi Sekretariat dapat memberikan data term dan jadwal kelas ke Aplikasi lain melalui REST service.

Beberapa **contoh API yang mungkin dimiliki** oleh Aplikasi Sekretariat dan dapat diakses aplikasi lain:

- **/api/getAllKelas**
  Mengembalikan daftar seluruh kelas yang bisa diambil oleh mahasiswa dalam pengisian IRS pada term tertentu

- **/api/getStudentKurikulum**
  Mengembalikan data kurikulum yang digunakan oleh seorang mahasiswa.

- **/api/getTerm**
  Mengembalikan data term.

Aplikasi Sekretariat membutuhkan data dari **Aplikasi Kurikulum** untuk melakukan pengaturan kurikulum suatu angkatan pada suatu prodi. Aplikasi Sekretariat juga membutuhkan data mahasiswa, data riwayat perkuliahan mahasiswa, dan data kurikulum dari **Aplikasi Universitas**, **Aplikasi Kurikulum, dan Aplikasi Akademik Mahasiswa** untuk menampilkan jumlah mahasiswa yang lulus dari setiap prodi dan angkatan. **Aplikasi Sekretariat mungkin membutuhkan data dari Aplikasi lain.**

Anda dapat menyesuaikan API yang akan dibuat sesuai dengan kebutuhan dan kesepakatan dengan kelompok lain.

Aplikasi Akademik Mahasiswa
Aplikasi Akademik Mahasiswa digunakan untuk melakukan pengisian IRS dan menampilkan riwayat perkuliahan mahasiswa. Aplikasi Akademik Mahasiswa juga dapat menampilkan status yudisium mahasiswa, apakah mahasiswa sudah berhak lulus atau belum.
Aplikasi Akademik Mahasiswa memiliki kapabilitas untuk:

- **Mengelola Mahasiswa, CRUD**
- **Mengelola IRS mahasiswa, CRUD**
  Hanya bisa mengisi IRS jika pada periode pengisian IRS saja pada term yang telah ditentukan oleh Aplikasi Sekretariat.

- **Menampilkan dashboard riwayat perkuliahan mahasiswa (butuh API Aplikasi Penilaian dan API Aplikasi Sekretariat)**
  Menampilkan riwayat perkuliahan mahasiswa termasuk nilai mata kuliah, IP, IPK, SKS lulus, dll.

- **Menampilkan status kelulusan (yudisium) mahasiswa (butuh API Aplikasi Penilaian dan Aplikasi Kurikulum)**
  Menampilkan status apakah mahasiswa sudah boleh yudisium atau belum.

Aplikasi Akademik Mahasiswa hanya menyimpan data:
- Mahasiswa
- Isian IRS Mahasiswa
- Riwayat perkuliahan mahasiswa

Aplikasi Akademik Mahasiswa dapat memberikan data IRS mahasiswa dan riwayat perkuliahan mahasiswa ke Aplikasi lain melalui REST service.

Beberapa **contoh API yang mungkin dimiliki** oleh Aplikasi Akademik Mahasiswa dan dapat diakses aplikasi lain:
- **/api/getRiwayatKuliah**
  Mengembalikan riwayat perkuliahan seorang mahasiswa

- **/api/getIRS**
  Mengembalikan isian IRS seorang mahasiswa

Aplikasi Akademik Mahasiswa membutuhkan data kurikulum dari **Aplikasi Kurikulum** untuk mengetahui status kelulusan mahasiswa. Selain itu, juga dibutuhkan data daftar kelas yang ditawarkan untuk *term* tertentu kepada mahasiswa dalam pengisian IRS dari Aplikasi Sekretariat. **Aplikasi Akademik Mahasiswa mungkin membutuhkan data dari Aplikasi lain.**

Anda dapat menyesuaikan API yang akan dibuat sesuai dengan kebutuhan dan kesepakatan dengan kelompok lain.

Aplikasi Penilaian
Aplikasi Penilaian digunakan oleh dosen untuk melakukan penilaian terhadap mahasiswa.
Aplikasi Penilaian memiliki kapabilitas untuk:
- **Menampilkan daftar peserta dan nilai peserta suatu mata kuliah (butuh API Aplikasi Akademik Mahasiswa, API Aplikasi Sekretariat, dan API Aplikasi Universitas)**
  Asumsikan dosen memiliki akses untuk melihat seluruh mata kuliah dari suatu prodi. Tampilkan daftar nilai dari seluruh mahasiswa dari mata kuliah tersebut.

- **Melakukan pengisian nilai untuk setiap mahasiswa peserta pada suatu mata kuliah (butuh API Aplikasi Akademik Mahasiswa dan API Aplikasi Sekretariat)**
  Pengisian nilai dilakukan dengan nilai angka.

- **Menampilkan statistik seperti mahasiswa dengan nilai tertinggi, mahasiswa dengan nilai terendah, nilai rata-rata, standard deviation dari sebuah mata kuliah (butuh API Aplikasi Akademik Mahasiswa)**
  Setiap mata kuliah memiliki statistik nilai akhir tertinggi, nilai akhir terendah, rata-rata nilai akhir, dan standar deviasi nilai akhir. Tampilkan pula mahasiswa dengan nilai akhir tertinggi dan terendah tersebut.

Aplikasi Penilaian hanya menyimpan data:
- Nilai mahasiswa untuk setiap mata kuliah
- Statistik nilai mata kuliah

Beberapa **contoh API yang mungkin dimiliki** oleh Aplikasi Penilaian dan dapat diakses aplikasi lain:
- **/api/getNilaiKuliah**
  Mengembalikan nilai suatu mata kuliah dari seorang mahasiswa

- **/api/getDaftarNilaiKuliah**
  Mengembalikan daftar peserta dan nilai suatu mata kuliah

- **/api/getStatistikNilaiKuliah**
  Mengembalikan statistik nilai dari suatu mata kuliah [Revisi 1.1]

Aplikasi Penilaian dapat memberikan data nilai mahasiswa ke Aplikasi lain melalui REST service.

Aplikasi Penilaian membutuhkan data dari **Aplikasi Akademik Mahasiswa** untuk data mahasiswa dan isian IRS. Aplikasi Penilaian juga membutuhkan data mata kuliah dari **Aplikasi Sekretariat**. **Aplikasi penilaian mungkin membutuhkan data dari Aplikasi lain juga.**

Anda dapat menyesuaikan API yang akan dibuat sesuai dengan kebutuhan dan kesepakatan dengan kelompok lain.

Bonus Portal Aplikasi

Bonus akan diberikan kepada kelompok besar jika mengimplementasikan sebuah project baru, yaitu Portal Aplikasi. Portal Aplikasi merupakan pintu masuk bagi pengguna untuk menggunakan kumpulan Aplikasi diatas. Contoh Portal Aplikasi adalah http://apps.cs.ui.ac.id. Portal Aplikasi dikerjakan di project berbeda. Portal Aplikasi boleh dikerjakan oleh siapapun anggota dari kelompok besar.

Fitur Login

Setiap Aplikasi perlu mengimplementasi fitur login masing-masing untuk pengguna dari masing-masing aplikasi tersebut. Perlu diperhatikan masalah keamanan Aplikasi, yaitu ada *route* yang perlu otentikasi untuk diakses dan ada *route* yang tidak.

Fitur Diluar Proses Bisnis

Selain fitur yang termasuk proses bisnis diatas, Anda dan tim diminta untuk memenuhi fitur dibawah demi menciptakan sistem yang terpercaya dan dapat diandalkan. Fitur tersebut antara lain:

- **Validasi API**
  Validasi perlu dilakukan untuk semua REST service untuk menjamin setiap *service* konsisten dalam mengembalikan keluaran. Contohnya, jika *request* memberikan *payload* yang salah maka service juga perlu mengembalikan *response* ynag memberitahukan bahwa input tidak benar atau tidak sesuai dengan spesifikasi. Jika *request* sesuai maka *service* perlu memberikan *response* sesuai dengan permintaan *request*.

  Pada kode Anda, Anda perlu melakukan validasi *request* yang dilakukan oleh kelompok lain. Contoh validasi *request* misalkan pada API 'kurikulum/api/getKurikulum' diperlukan parameter id_kurikulum dalam *query string*-nya. Jika tidak disertakan maka API Anda akan mengembalikan JSON kosong saja. Kelompok yang melakukan *request* juga perlu melakukan validasi apakah kembalian *response* dari API yang dituju berisi data yang diinginkan atau kosong.

  Beberapa hal yang perlu Anda perhatikan misalkan *request method* (POST atau GET), JSON *payload*/*path variable*/*query string*. Anda dibebaskan untuk menentukan metode POST atau GET, Anda tentukan sendiri, untuk mengirim *request*. Anda dan tim bisa mencari tahu lebih lanjut lagi tentang topik**Dokumentasi API** di sumber lain.

- ***User Interface* yang selaras untuk semua aplikasi**
  Untuk meningkatkan kepercayaan pengguna, Anda perlu menampilkan *user interface* yang selaras untuk seluruh aplikasi. Hal ini perlu dilakukan agar semua pengguna mendapatkan pengalaman yang sama dan memberikan informasi secara implisit bahwa

semua aplikasi tersebut merupakan satu kesatuan. Keselarasan *user interface* juga meningkatkan kepercayaan pengguna dalam menggunakan sistem. Anda dan tim besar Anda perlu mendiskusikan perancangan *user interface* tersebut.

- **Kami merekomendasikan Anda menggunakan Bootstrap, jQuery, dan DataTables.**
  Kami merekomendasikan Anda menggunakan bootstrap, jQuery, dan DataTables untuk kemudahan Anda. Walaupun begitu Anda dibebaskan untuk menggunakan library lain untuk menunjang fungsionalitas yang Anda kembangkan.

- **Secara ideal, Anda dan tim tidak perlu tahu implementasi dari kelompok kecil lain. Cukup melihat dokumentasi API kelompok kecil lain untuk mengetahui API apa saja yang bisa dipanggil.**
  Jika semua kelompok di kelompok besar telah mendefinisikan dokumentasi API masing-masing, maka kelompok Anda tidak perlu lagi mengetahui implementasi dari kelompok lain karena apapun implementasi yang dilakukan kelompok lain kelompok tersebut perlu memberikan respon jika diberikan *request* sesuai dengan dokumentasi API yang mereka buat.

Asumsi, Limitasi, dan Penyesuaian

Berikut beberapa asumsi dan limitasi yang perlu Anda ketahui dan pahami:

- **Anda dan tim diperbolehkan untuk menambahkan asumsi dan fitur lainnya untuk Aplikasi Anda**
  Spesifikasi fitur yang diminta dari masing-masing Aplikasi masih dapat dikembangkan dan digali lebih dalam. Anda perlu menambahkan **asumsi** dan **analisis** Anda untuk setiap fitur. Anda dapat menambahkan asumsi dan fitur lainnya untuk dari hasil diskusi dengan kelompok kecil dan kelompok besar Anda. Lakukan observasi pada sistem dan kasus nyata SIAK-NG sebagai sumber analisis.

- **Setiap *endpoint* dari *web service* tidak perlu dilindungi dengan otentikasi**
  Aplikasi lain, atau client lain, dapat mengakses *web service* tanpa login atau otentikasi ke Aplikasi yang diakses. Namun, pengguna setiap aplikasi tetap perlu login untuk mengakses semua fitur. Hal ini untuk memudahkan pengecekan dan testing.

- **Anda HARUS mengakses API kelompok kecil lain dan memproses data dari API tersebut di *controller* Spring Boot**
  Tanpa bermaksud membatasi kebebasan Anda dalam melakukan pengembangan dan menilai tingkat pemahaman Anda terhadap materi yang diberikan, kami mengharuskan Anda untuk mengakses dan memproses data dari API kelompok lain melalui controller Spring Boot. Anda **tidak diperkenankan** menggunakan *library* atau *framework* seperti

ReactJS, Vue.js, Angular, jQuery, atau Vanilla JavaScript dan sejenisnya untuk mengakses **API kelompok lain.** Anda tetap dapat menggunakan *library* atau *framework* tersebut untuk mengakses API yang Anda buat khusus hanya untuk konsumsi Aplikasi Anda sendiri.

- Untuk kelompok kecil Aplikasi Akademik Mahasiswa yang berada di kelompok besar yang hanya memiliki 3 kelompok kecil dapat menyimpan sendiri nilai angka mata kuliah mahasiswa pada data riwayat perkuliahan mahasiswa. Nilai tersebut sudah terisi di database. **Tidak perlu membuat fitur mengisi nilai. Jika ingin melakukan perubahan lakukan lewat database langsung saja.**

Pembagian Kelompok

Pembagian kelompok silakan mengacu ke pengumuman pada Scele APAP.

Penilaian

Penilaian kelompok kecil akan dibagi menjadi beberapa komponen, yaitu:

- Desain arsitektur *layer* (Service, DAO, Controller, dll) masing-masing Aplikasi (30 poin)
- Fitur masing-masing Aplikasi, baik yang membutuhkan data dari Aplikasi lain maupun tidak (30 poin)
- Desain *web service* masing-masing Aplikasi (30 poin)
- *User interface* dan interaksi (10 poin)

Penilaian dilakukan oleh tim dosen dan tim asisten dengan sistem penilaian berbeda dengan tutorial atau tugas. Penilaian akan dilakukan secara individu dan kelompok. Kontribusi dari masing-masing individu akan sangat dihargai dalam tugas kelompok ini. Pada presentasi akhir, Anda perlu mengisi borang *peer review* untuk masing-masing individu kelompok kecil terkait kontribusi mereka masing-masing. **Nilai individu bisa berbeda dari nilai kelompok tergantung dari *peer review.***

Hal yang perlu Anda perhatikan adalah kerja sama antar kelompok kecil dalam kelompok besar. Beberapa fitur saling berkaitan satu sama lain. Oleh karena itu, sebuah kelompok kecil juga ikut bertanggung jawab atas fitur kelompok kecil lainnya. Pastikan bahwa kelompok kecil Anda dapat menyelesaikan semua fitur yang ada sehingga tidak menyulitkan kelompok kecil lain dalam membangun fitur.

Tips dan Trik Pengerjaan

Berikut tips dan trik pengerjaan tugas akhir:

- **Melakukan analisis fitur dan basis data terlebih dahulu sebelum terjun ke kode**
  Mulai pengerjaan tugas akhir dengan melakukan analisis fitur dan basis data terlebih dahulu, sehingga ketika Anda mulai membuat kode semua fitur sudah ditetapkan dan didesain dengan baik dan tinggal dituangkan dalam kode. Desain dan Aplikasi yang buruk pada tahap awal pengerjaan proyek dapat mengganggu pengerjaan kode tugas ditengah-tengah karena perlu ada perubahan. Hal tersebut dapat mempengaruhi *pace* kelompok Anda dalam merampungkan kode tugas kelompok Anda.

- **Lakukan diskusi antar kelompok kecil untuk menyepakati API Contract**
  Lakukan diskusi untuk menyepakati API Contract antar kelompok kecil. Elemen paling penting dari API Contract adalah *request* dan *response*. Antar kelompok kecil perlu membuat kesepakatan *request* yang akan dibuat dan seperti apa struktur respon yang dikembalikan.

- **Buat dokumentasi API dari masing-masing service sejak awal**
  Buat API *contract* untuk setiap fitur Aplikasi Anda sejak awal setelah melakukan analisis fitur dan basis data. Jika semua kelompok kecil telah membuat API *contract* masing-masing, setiap kelompok dapat mengerjakan fitur masing-masing secara paralel karena berpatokan dari API Contract yang sudah disepakati.

- **Buat *Service Consumer* untuk melakukan *testing* dengan data *dummy***
  Agar tidak saling menunggu pembuatan API antar kelompok kecil, Anda dapat membuat sebuah *project* lain sebagai *Service Producer* di *port* berbeda untuk mengembalikan data *dummy* sesuai dengan dokumentasi API yang telah disepakati. Dengan begitu, Anda dapat melakukan *testing* tanpa harus menunggu fitur kelompok lain selesai.

- **Anda dapat membuat *web service* yang digunakan sendiri**
  Anda juga dipersilakan untuk membuat *web service* khusus kelompok kecil yang Anda gunakan sendiri, API tersebut tidak diakses kelompok lain. Anda dapat bereksplorasi disini bagaimana menggunakan API dan mengakses API tersebut melalui, misalkan, DataTables atau jQuery di front-end Aplikasi Anda untuk meningkatkan pengalaman dan kenyamanan pengguna dalam berinteraksi dengan Aplikasi Anda.

File Pendukung

Anda dapat menggunakan beberapa file pendukung untuk membantu Anda dalam mengerjakan tugas akhir ini. Beberapa file pendukung tersebut, yaitu:

- Contoh dokumentasi API pada repository Aplikasi Universitas [Revisi 1.1]

Sesuaikan penggunaan file-file tersebut untuk membantu tugas akhir Anda. File pendukung tersebut dapat Anda akses di GitHub https://github.com/apap-2017/univ-app [Revisi 1.1]

Deliverables

Deliverables untuk tugas akhir kali ini adalah:

1. ***File Project***
   Buat sebuah private [Revisi 1.1] project baru pada *organization* /apap-2017 dengan format nama **TA_KODE-KELOMPOK-KECIL**, contoh **TA_A2** untuk Kelompok besar A kelompok kecil 2. *Push project* Anda ke *repository* GitHub tersebut. Pastikan Anda tidak salah dalam melakukan *push repository* dan tidak melebihi batas waktu yang telah ditentukan!

   Commit yang akan diperhitungkan adalah ***commit* terakhir di *branch master*** dari salah satu anggota kelompok tersebut.

   Perlu diperhatikan bahwa setiap mahasiswa perlu memiliki *commit history* ke repository kelompok kecil. Setiap mahasiswa perlu rutin *commit* dan *push* setelah mengerjakan fitur. *Commit* hanya sekali saja saat *deadline* **tidak diperkenankan** dan akan mempengaruhi penilaian individu. Untuk skala tugas akhir seperti ini, seharusnya setiap individu memiliki jumlah *commit* yang cukup banyak jika kontribusi setiap individu merata di kelompok kecil.

   Catatan: Setiap kelompok kecil dalam satu kelompok besar diperbolehkan mengakses dan pull repository kelompok kecil lain untuk keperluan testing. [Revisi 1.1]

2. **Dokumentasi API**
   Dokumentasi API berisi daftar API/REST Service yang dapat diakses.
   Buatlah file **readme.md** pada *root repository* tugas akhir. Tuliskan dokumentasi API Anda pada *file* readme.md. Contoh penulisan dokumentasi API bisa Anda dapatkan di *repository* pendukung yang telah dijelaskan diatas. Walaupun sudah kami berikan contoh, Anda diberi kebebasan dalam menuliskan format dokumentasi API asalkan jelas dan dapat dipahami.

3. ***Write-up***
   Buat sebuah *file write-up* untuk sebuah kelompok kecil, bukan per orang. Jelaskan **apa saja hal yang Anda pelajari** dari tugas akhir ini dan juga **ceritakan proses pengerjaan termasuk kendala dan tantangan yang Anda dan tim hadapi**!

Kami rekomendasikan Anda menggunakan format pdf agar dapat lebih leluasa dalam menuangkan penjelasan Anda dengan dukungan *screenshot* dan kode. Masukkan *file writeup* ke *folder project*. Pastikan *file write-up* juga di-*push* ke *repository*.

Deadline

Deadline progress:

- **Rancangan Basis data, User Interface, dan Dokumentasi API**
  Submisi Scele: 17 November 2017, 23:55:00
  Format penamaan:
  TA_ProgressDesain_[KODE-KELOMPOK-KECIL].pdf,
  Contoh: TA_ProgressDesain_A6
  Buat sebuah dokumen yang berisi **rancangan** API, **rancangan** basis data yang akan digunakan, dan **rancangan** *user interface* (*low/high-fidelity* atau tampilan HTML). Anda dipersilakan menambahkan poin lain yang perlu diketahui asisten dan dosen. Kemudian tim asisten akan memberikan *feedback* dari submisi yang diberikan sebagai saran perbaikan. Hanya perlu dikumpulkan oleh salah satu anggota kelompok kecil saja.

- **Laporan Progress Pengembangan**
  Submisi Scele: 24 November 2017, 23:55:00
  Format penamaan:
  TA_Progress_[KODE-KELOMPOK-KECIL].pdf,
  Contoh: TA_Progress_A6
  Siapkan laporan sederhana untuk menyampaikan progress pengerjaan tugas. Laporan berisi daftar fitur, status pengerjaan, dan kendala/tantangan pengerjaan. Hanya perlu dikumpulkan oleh salah satu anggota kelompok kecil saja.

Deadline final:
Deadline final terdiri dari laporan akhir dan source code. Source code pada GitHub memiliki commit terakhir sebelum waktu deadline. **Laporan akhir di submit melalui Scele.** Laporan akhir berisi tentang:

- Daftar fitur dan Dokumentasi API yang dikembangkan
- Penjelasan: Apa saja yang Anda dan tim pelajari dalam pengerjaan tugas akhir
- Penjelasan:Apa kendala dan tantangan yang Anda dan tim hadapi dalam pengerjaan tugas akhir
- Komentar dan saran tentang Tugas Akhir APAP

Format penamaan: TA_Final_[KODE-KELOMPOK-KECIL].pdf, Contoh: TA_Final_A6
Deadline Final: 7 Desember 2017, 23:59:59

Presentasi dan demo final:

- TBA

Penalti

Penalti:

- **Keterlambatan pengumpulan *deadline* Rancangan Basis data, User Interface, dan Dokumentasi API**
  Penalti sebesar -5 poin akan ditambahkan ke nilai akhir kelompok

- **Keterlambatan pengumpulan *deadline* Laporan Progress Pengembangan**
  Penalti sebesar -5 poin akan ditambahkan ke nilai akhir kelompok

- **Keterlambatan pengumpulan *deadline final***
  Penalti sebesar -10 poin akan ditambahkan ke nilai akhir kelompok