«SKRIPSI/TUGAS AKHIR»

«JUDUL BAHASA INDONESIA»

«Nama Lengkap»

NPM: «10 digit NPM UNPAR»

PROGRAM STUDI «MATEMATIKA/FISIKA/TEKNIK INFORMATIKA»
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
«tahun»

«FINAL PROJECT/UNDERGRADUATE THESIS»

«JUDUL BAHASA INGGRIS»

«Nama Lengkap»

NPM: «10 digit NPM UNPAR»

DEPARTMENT OF «MATHEMATICS/PHYSICS/INFORMATICS»
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
«tahun»

LEMBAR PENGESAHAN

«JUDUL BAHASA INDONESIA»

«Nama Lengkap»

NPM: «10 digit NPM UNPAR»

Bandung, «tanggal» «bulan» «tahun»

Menyetujui,

Pembimbing Utama	Pembimbing Pendamping
«pembimbing utama/1»	«pembimbing pendamping/2»

Ketua Tim Penguji	Anggota Tim Penguji
«penguji 1»	«penguji 2»

Mengetahui,

Ketua Program Studi

«Ketua Program Studi»

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa «skripsi/tugas akhir» dengan judul:

«JUDUL BAHASA INDONESIA»

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal «tanggal» «bulan» «tahun»

Meterai
Rp. 6000

«Nama Lengkap»
NPM: «10 digit NPM UNPAR»

ABSTRAK

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia»
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Kata-kata kunci: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Indonesia»

ABSTRACT

«Tuliskan abstrak anda di sini, dalam bahasa Inggris»
Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Keywords: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Inggris»

«kepada siapa anda mempersembahkan skripsi ini. . . ?»

KATA PENGANTAR

«Tuliskan kata pengantar dari anda di sini . . . »
Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Bandung, «bulan» «tahun»

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Outlook.com[1] adalah sebuah kumpulan aplikasi berbasis *web* seperti *webmail*, *contacts*, *tasks*, dan *calendar* dari *Microsoft*. Fitur calendar sendiri pertama dirilis pada 14 Januari 2008 dengan nama Windows Live Calendar. Fitur calendar yang dimiliki oleh *Outlook.com Calendar* sendiri memiliki tampilan yang mirip dengan aplikasi kalender *desktop* pada umumnya. Seperti layaknya kalender digital pada umumnya, aplikasi *Outlook.com Calendar* juga bisa menambahkan, menyimpan, dan memodifikasi *event-event* yang dimasukkan oleh pengguna dan bisa dibuka dimana saja karena bersifat *online*.

Slack[2] adalah alat dan layanan kolaborasi tim berbasis *cloud*. *Slack* merupakan singkatan dari "*Searchable Log of All Conversation and Knowledge*". Cara melakukan kolaborasi di aplikasi *Slack* sendiri adalah dengan komunitas, grup, atau tim bergabunga ke dalam URL yang spesifik. *Room chat* yang terdapat di dalam aplikasi *Slack* biasa disebut dengan *Channel*. Ada 2 jenis *channel* di dalam aplikasi *Slack* yaitu *Public Channel* dan *Private Channel*. Pada *Public Channel*, seluruh anggota dari tim atau komunitas bisa masuk dan bergabung untuk berkomunikasi di *channel* tersebut. Tetapi pada *Private Channel*, hanya anggota yang diizinkan, ditambahkan, dan diundang oleh admin atau pembuat *channel* sajalah yang bisa ikut serta dalam berkomunikasi di dalam *channel* tersebut. *Slack* juga terintegrasi dengan banyak layanan pihak ketiga seperti contohnya adalah *Google Drive*, *Github*, *Trello*, *Dropbox*, dan masih banyak lagi layanan pihak ketiga yang bisa diintegrasikan dengan *Slack* itu sendiri.

Pada *Slack* terdapat status pengguna yang bisa diganti oleh pengguna tersebut untuk menggambarkan keadaan pengguna saat ini. Sebagai *default*, status bisa menggambarkan jika pengguna sedang "*In a meeting*" atau sedang "*Out Sick*", dan banyak status default yang disediakan oleh *Slack*. Serta status pun bisa diisi oleh pengguna secara sendiri sesuai dengan apa yang ingin dituliskan oleh penggunanya. Disinilah yang menjadi latar belakang dirancangnya perangkat lunak ini yaitu terkadang pengguna lupa untuk mengganti status menjadi "*In a meeting*" saat pengguna memiliki jadwal untuk melakukan meeting, sehingga status di pengguna masih terlihat tersedia oleh user lain yang membuat tidak mengetahui sang pengguna sedang dalam keadaan *meeting* yang tidak dapat diganggu. Di saat seperti ini, kemungkinan untuk *meeting* terganggu oleh adanya *chat* yang masuk lewat *Slack* pun cukup tinggi.

Pada skripsi ini akan dibuat perangkat lunak yang akan membaca jadwal dari pengguna yang dicantumkan di aplikasi *Outlook.com Calendar*, lalu akan di integrasikan kepada aplikasi *Slack* dengan mengubah dan mengganti status sesuai dengan jadwal yang telah didapatkan dari data di *Outlook.com Calendar* dari pengguna.

Perangkat lunak ini akan dibuat menggunakan *Node.js*[3] dan akan memiliki 2 fungsi utama yaitu yang pertama adalah membaca dan mencatat jadwal dari *Outlook.com Calendar* yang membutuhkan adanya *Outlook.com Calendar API*. Lalu perangkat lunak ini juga memiliki fungsi kedua yaitu

[1] https://outlook.live.com/
[2] https://slack.com/
[3] https://nodejs.org

mengubah status ke aplikasi *Slack* dengan menggunakan *Slack API*. Nantinya kedua fungsi dari perangkat lunak ini akan dijalankan secara berkala.

1.2 Rumusan Masalah

Pada perangkat lunak ini, terdapat rumusan masalah sebagai berikut:

1. Bagaimana cara mendapatkan data *event* dari *Outlook.com Calendar*?
2. Bagaimana mengubah status pada aplikasi *Slack* menggunakan *Slack* API?
3. Bagaimana cara membuat program agar dapat mengubah status pada aplikasi *Slack* di jadwal yang telah didapat dari aplikasi *Outlook.com Calendar*?

1.3 Tujuan

Adapun pada perangkat lunak ini memiliki tujuan sebagai berikut:

1. Mengetahui cara mendapatkan data *event* dari *Outlook.com Calendar*.
2. Mengetahui cara mengubah status pada aplikasi *Slack* menggunakan *Slack* API.
3. Membuat program agar dapat mengubah status pada aplikasi *Slack* di jadwal yang telah didapat dari aplikasi *Outlook.com Calendar*.

1.4 Batasan Masalah

Perancangan perangkat lunak ini dibuat berdasarkan batasan-batasan sebagai berikut:

1. Program ini dijalankan secara berkala sehingga tidak dapat menjalankan *update* status secara *real-time*.

1.5 Metodologi

Berikut adalah metodologi yang akan digunakan dalam penelitian ini:

1. Melakukan studi literatur tentang *Outlook.com Calendar*, *Slack*, dan juga *Node.js*.
2. Menggunakan aplikasi *Slack* di lingkungan tempat penulis melakukan magang.
3. Menganalisis aplikasi-aplikasi sejenis.
4. Melakukan analisis cara melakukan *synchronize* dengan aplikasi *Outlook.com Calendar* secara berkala.
5. Merancang bagian dari perangkat lunak yang akan mengambil data-data *event* dari *Outlook.com Calendar* dan yang bertugas untuk mengubah status pada *Slack* saat waktu sesuai dengan jadwal yang sudah tercatat dari *Outlook.com Calendar*.
6. Mengimplementasi bagian pengambilan data dari *Outlook.com Calendar* dan juga bagian mengatur status pada *Slack* sesuai jadwal yang telah diambil kepada perangkat lunak Integrasi *Outlook.com Calendar* dengan *Slack* serta melakukan pengujian terhadap fitur yang telah diimplementasikan.

1.6 Sistematika Pembahasan

Setiap bab dalam penelitian ini akan memiliki sistematika pembahasan yang dijelaskan ke dalam poin-poin sebagai berikut:

1. Bab 1: Pendahuluan, yaitu menjelaskan gambaran umum dari penelitian ini yang berisi tentang latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi, dan sistematika pembahasan.

2. Bab 2: Dasar Teori, yaitu menjelaskan dan membahas teori-teori yang dibutuhkan dan mendukung berjalannya penelitian ini. Meliputi tentang *Outlook.com Calendar API*, *Slack API*, *Node.js*, dan juga *Crontab*.

3. Bab 3: Analisis, yaitu membahas mengenai analisis masalah. Berisi tentang analisis aplikasi sejenis, analisis cara pengambilan data dari *Outlook.com Calendar*, dan proses pengubahan status menggunakan program pada *Slack*.

4. Bab 4: Perancangan, yaitu membahas mengenai perancangan aplikasi untuk melakukan sinkronisasi antara *Outlook.com Calendar* dengan *Slack*.

5. Bab 5: Implementasi dan Pengujian, yaitu membahas mengenai implementasi dari aplikasi yang telah dirancang dan juga pengujian aplikasi tersebut.

6. Bab 6: Kesimpulan dan Saran, yaitu berisi tentang kesimpulan dari penelitian ini dan juga saran yang dapat diberikan untuk penelitian selanjutnya.

BAB 2

LANDASAN TEORI

2.1 *Microsoft Graph API*

Microsoft Graph API adalah webservice yang berguna untuk mendapatkan data-data yang terdapat di dalam layanan Microsoft 365 yaitu seperti *Azure Active Directory*, layanan *Office* 365 (*SharePoint, OneDrive, Outlook/Exchange, Microsoft Teams, OneNote, Planner,* dan *Excel*), layanan *Enterprise Mobility and Security* (*Identity Manager, Intune, Advanced Threat Analytics,* dan *Advanced Threat Protection*), layanan *Windows* 10 (*activities* dan *devices*), dan *Education.* Terdapat 2 versi referensi untuk *Microsoft Graph API* yaitu versi 1.0 dan juga versi beta, tetapi yang dituliskan pada subbab ini mengacu kepada versi 1.0. Pada versi 1.0, *endpoint* utama yang dipakai adalah mengacu kepada *endpoint https://graph.microsoft.com/v1.0.* Untuk menggunakan fungsi dari *Microsoft Graph API*, dibutuhkan untuk mendaftarkan terlebih dahulu aplikasi yang akan dirancang dan memakai fungsi dari *webservice* dari *Microsoft Graph API* ke *Microsoft App Registration Portal*[1]. Pada saat mendaftarkan aplikasinya, pastikan untuk menyalin dan menyimpan *application ID* yang adalah pengenal unik untuk aplikasi yang didaftarkan, dan juga menyalin *Redirect URL* yang didaftarkan sebagai *URL* yang akan menerima balikan *authentication* dan juga *token* yang akan dikirim oleh *endpoint Azure AD v2.0*, serta menyalin *application secret* yang didapat saat mengklik "*Generate New Password*" saat mendaftarkan aplikasi (berlaku jika mendaftarkan aplikasi berjenis *web apps*). *Application ID* yang didapat saat mendaftar aplikasi akan dipakai untuk mengisi nilai dari parameter *client_id* yang akan diisi saat akan melakukan *request* untuk mendapatkan *authorization_code.* Setelah mendapatkan *authorization_code*, maka langkah selanjutnya adalah meminta *access_token* yang membutuhkan *parameter authorization_code* kepada *field code*, dan juga *application_secret* yang didapat dari pendaftaran aplikasi sebelumnya yang akan mengisi *field client_secret.* Response dari request token akan mengembalikkan jangka waktu aktif dari token tersebut dan juga refresh_token yang akan berguna untuk meminta refresh_token saat token sudah *expired.*

Setelah mendapatkan access token, barulah layanan untuk mendapatkan data yang tersimpan di Microsoft baru bisa diakses dan didapatkan. Ada banyak layanan yang disediakan dari Microsoft Graph API

2.1.1 User resource type

Kelas user ini merepresentasikan Azure AD user account yang memiliki method-method:

- **List users** Method ini berfungsi untuk mendapatkan daftar dari objek pengguna.
- **Create user** Method ini berfungsi untuk membuat pengguna.
- **Get user** Method ini berfungsi untuk membaca properti dan juga hubungan pengguna.
- **Update user** Method ini berfungsi untuk memperbaharui pengguna.

[1]https://apps.dev.microsoft.com/

- **Delete user** Method ini berfungsi untuk menghapus pengguna.
- **List messages** Method ini berfungsi untuk mendapatkan semua pesan di kotak surat pengguna yang masuk.
- **Create message** Method ini berfungsi untuk membuat pesan baru untuk dimasukkan ke dalam koleksi pesan.
- **List mailFolders** Method ini berfungsi untuk mendapatkan folder-folder surat dibawah folder root dari pengguna yang masuk.
- **Create mailFolder** Method ini berfungsi untuk membuat mailFolder ke dalam koleksi mailFolders.
- **sendMail** Method ini berfungsi untuk mengirim pesan.
- **List events** Method ini berfungsi untuk mendapatkan daftar objek event di dalam kotak pesan dari pengguna.
- **Create event** Method ini berfungsi untuk membuat event ke dalam koleksi dari event-event.
- **List calendars** Method ini berfungsi untuk mendapatkan daftar objek calendar.
- **Create calendar** Method ini berfungsi untuk membuat objek calendar baru yang akan dikirim ke dalam koleksi objek calendars.
- **List calendarGroups** Method ini berfungsi untuk mendapatkan daftar objek calendarGroup.
- **Create calendarGroup** Method ini berfungsi untuk membuat objek calendarGroup baru kedalam koleksi calendarGroup.
- **List calendarView** Method ini berfungsi untuk mendapatkan koleksi objek event.
- **List contacs** Method ini berfungsi untuk mendapatkan daftar kontak dari folder Contacts pengguna yang masuk.
- **Create contact** Method ini berfungsi untuk membuat kontak baru untuk dimasukkan ke dalam koleksi kontak.
- **List contactFolders** Method ini berfungsi untuk mendapatkan koleksi folder kontak dari pengguna yang masuk.
- **Create contactFolder** Method ini berfungsi untuk membuat folder kontak.
- **List directReports** Method ini berfungsi untuk mendapatkan pengguna dan kontak yang melaporkan pengguna dari properti directReports.
- **List manager** Method ini berfungsi untuk mendapatkan manager pengguna dari properti manager.
- **List memberOf** Method ini berfungsi untuk mendapatkan kelompok dan peran dari anggota langsung pengguna lewat properti memberOf.
- **List transitive memberOf** Method ini berfungsi untuk mendapatkan kelompok dan peran dari pengguna lewat properti memberOf, tetapi method ini bersifat transitif dan mencakup grup-grup dimana pengguna menjadi anggota.
- **List ownedDevices** Method ini berfungsi untuk mendapatkan perangkat yang dimiliki oleh pengguna dari properti ownedDevices.

- **List ownedObjects** Method ini berfungsi untuk mendapatkan objek yang dimiliki pengguna yang didapat dari properti ownedObjects.
- **List registeredDevices** Method ini berfungsi untuk mendapatkan perangkat yang terregistrasi oleh pengguna dari properti registeredDevices.
- **List createdObjects** Method ini berfungsi untuk mendapatkan objek yang dibuat oleh pengguna dari properti createdObjects.
- **assignLicense** Method ini berfungsi untuk menambah atau membuang "subscriptions" dari pengguna, serta bisa untuk mengaktifkan dan menonaktifkan paket spesifik terkait dengan langganan.
- **List licenseDetails** Method ini berfungsi untuk mendapatkan koleksi objek licenseDetails.
- **checkMemberGroups** Method ini berfungsi untuk memeriksa keanggotaan dalam daftar grup.
- **getMemberGroups** Method ini mengembalikkan semua grup dimana pengguna menjadi anggota didalamnya.
- **getMemberObjects** Method ini mengembalikkan semua grup dan peran dimana pengguna menjadi anggota didalamnya.
- **reminderView** Method ini mengembalikkan daftar pengingat di kalender dengan jam mulai dan berakhirnya secara spesifik.
- **delta** Method ini berfungsi untuk mendapatkan perubahan tambahan pengguna.

2.2 *Outlook Calendar API*

Untuk mengakses dan mendapatkan data yang terdapat di *Outlook.com Calendar*, dibutuhkan *Microsoft Graph API*. *Microsoft Graph* itu sendiri adalah gerbang untuk mendapatkan data-data yang terdapat di *Microsoft* 365 dan *Outlook.com Calendar* termasuk di dalam layanan *Office* 365. *Microsoft Graph API* adalah sebuah *webservice* yang memiliki satu buah *endpoint* untuk memperoleh data yang ada pada *Azure Active Directory*, layanan *Office* 365(*SharePoint, OneDrive, Outlook/Exchange, Microsoft Teams, OneNote, Planner,* dan *Excel*), layanan *Enterprise Mobility and Security*(*Identity Manager, Intune, Advanced Threat Analytics,* dan *Advanced Threat Protection*), layanan *Windows* 10(*activities* dan *devices*), dan *Education* yaitu menuju *endpoint* ke *https://graph.microsoft.com*. *Microsoft Graph* dapat dipergunakan untuk membangun suatu aplikasi dengan memanfaatkan konteks unik dari masing-masing penggunanya. Aplikasi yang dapat dibangun dengan bantuan dari *Microsoft Graph* bisa jadi memiliki fungsi seperti:

- Melihat pertemuan/ *meeting* pengguna yang berikutnya sudah terjadwalkan serta membantu pengguna untuk mempersiapkan dengan memberikan informasi dari profil peserta.
- Melihat kepada kalender pengguna sehingga bisa merekomendasikan waktu yang tepat untuk pertemuan/ *meeting* selanjutnya.
- Mengubah file *Excel* yang berada di *OneDrive* pengguna secara *real-time* dan bisa melalui ponsel pengguna.
- Memantau perubahan yang ada di kalender pengguna sehingga bisa memngirimkan peringatan jika pengguna menghabiskan terlalu banyak waktu untuk melakukan rapat, serta bisa juga merekomendasikan *event-event* yang bisa didelegasikan atau dilewatkan berdasarkan dari seberapa relevannya *event* itu kepada pengguna.

- Membantu pengguna untuk bisa memilah informasi yang ditujukan untuk pekerjaan dengan informasi yang bersifat pribadi. Misalkan dengan pengelompokkan gambar-gambar pribadi yang masuk ke *OneDrive* pribadi serta memasukkan gambar-gambar pekerjaan ke *OneDrive for Business* pengguna.

Permintaan yang populer diminta pada *Microsoft Graph API* antara lain:

Operation	**URL**
GET my profile	https://graph.microsoft.com/v1.0/me
GET my files	https://graph.microsoft.com/v1.0/me/drive/root/children
GET my photo	https://graph.microsoft.com/v1.0/me/photo/$value
GET my mail	https://graph.microsoft.com/v1.0/me/messages
GET my high importance email	https://graph.microsoft.com/v1.0/me/messages?$filter=importance%20eq%20'high'
GET my calendar events	https://graph.microsoft.com/v1.0/me/events
GET my manager	https://graph.microsoft.com/v1.0/me/manager
GET last user to modify file foo.txt	https://graph.microsoft.com/v1.0/me/drive/root/children/foo.txt/lastModifiedByUser
GET Office365 groups I'm member of	https://graph.microsoft.com/v1.0/me/memberOf/$/microsoft.graph.group?$filter=groupTypes/any(a:a%20eq%20'unified')
GET users in my organization	https://graph.microsoft.com/v1.0/users
GET groups in my organization	https://graph.microsoft.com/v1.0/groups
GET people related to me	https://graph.microsoft.com/v1.0/me/people
GET items trending around me	https://graph.microsoft.com/beta/me/insights/trending
GET my notes	https://graph.microsoft.com/v1.0/me/onenote/notebooks

Dari *Microsoft Graph* ini, barulah dapat mengakses *Outlook.com Calendar API*. Cara untuk menggunakan membangun aplikasi menggunakan *API* dari *Microsoft* yang tersedia adalah:

Mendaftarkan aplikasi yang akan dibuat ke Azure AD.

Untuk Mendaftarkan aplikasi yang akan dibuat, ada beberapa langkah yaitu:

1. Masuk ke *Microsoft App Registration Portal* dengan menggunakan akun *Microsoft* atau akun sekolah atau akun kantor.
2. Pilih dan klik tombol bertuliskan "*Add an app*".
3. Masukkan nama untuk aplikasi yang akan dibuat dan klik tombol "*Create application*".
4. Salin "*application ID*". *Application ID* ini nantinya dipergunakan untuk mengkonfigurasikan aplikasinya.
5. Dibawah bagian "*Platforms*", pilih tombol "*Add Platforms*" dan pilihlah *platform* yang sesuai untuk aplikasi.

   Untuk *Native* atau *Mobile apps*:

   (a) Pilih "*Native Application*".

   (b) Salin bagian "*Built-in redirect URI*". Hal ini diperlukan untuk menkonfigurasi aplikasinya. *Redirect URI* ini disediakan identik untuk aplikasinya yang gunanya untuk memastikan bahwa pesan yang dikirim ke *URI* hanya dikirim ke aplikasi itu.

   Untuk *web apps*

   (a) Pilih "*Web*".

   (b) *Check* kotak dengan tulisan "*Allow Implicit Flow*" jika ingin mengaktifkan *OpenID Connect Hybrid* dan *implicit flow*. *Implicit Flow* memungkinkan aplikasi untuk menerima *sign-in info* dan juga *access token*, sedangkan nilai *default* dari bagian ini adalah *hybrid flow* dimana *flow* ini memungkinkan aplikasi untuk menerima *sign-in info* yaitu *token id*, dan juga *artifacts* atau dalam kasus ini adalah kode otorisasi yang digunakan aplikasi untuk mendapatkan *access token*.

   (c) Tentukan *redirect URI* yang adalah bagian dari aplikasi yang dihubungi oleh *endpoint* dari *Azure AD* 2.0 saat memproses permintaan otentikasi.

   (d) Dibawah "*Application Secrets*", pilihlah "*Generate New Password*" dan salinlah *app secret* yang terdapat di *New Password generated dialog box*. Perlu diketahui bahwa salinlah *app secret* sebelum *New password generated dialog* ditutup karena setelah ditutup, *app secret* tidak dapat diambil lagi.
6. Pilihlah "*Save*".

Tabel berikut menunjukkan *properties* yang perlu dikonfigurasi dan disalin untuk berbagai jenis aplikasi. Nilai *assigned.* berarti harus menggunakan nilai yang diberikan oleh *Azure AD*.

Mendapatkan otorisasi.

Langkah pertama untuk mendapatkan otorisasi adalah dengan cara meminta kepada *Azure AD* 2.0 **/authorize** *endpoint*. Azure ID akan mengecek pengguna yang masuk dan memastikan persetujuan pengguna untuk izin permintaan aplikasi tersebut. Berikut contoh *request* untuk mendapatkan *authorize code*:

```
https://login.microsoftonline.com/{tenant}/oauth2/v2.0/authorize?
client_id=6731de76-14a6-49ae-97bc-6eba6914391e
&response_type=code
&redirect_uri=http%3A%2F%2Flocalhost%2Fmyapp%2F
&response_mode=query
&scope=offline_access%20user.read%20mail.read
&state=12345
```

Parameter		**Deskripsi**
tenant	*required*	Nilai yang dipakai untuk mengontrol siapa yang bisa mengakses masuk ke aplikasi. Nilainya bisa diisi dengan akun *Microsoft*.
client_id	*required*	Diisi dengan *Application ID* yang diberikan oleh portal registrasi (*apps.dev.microsoft.com*).
response_type	*required*	Diisi dengan nilai "*code*" untuk menjalankan kode otorisasi.
redirect_uri	*recommended*	Diisi dengan redirect uri yang nantinya akan menerima respon dari otentikasi yang akan diterima oleh aplikasi. Nilai disini harus sama dengan yang sudah diisi di portal registrasi kecuali *URL* yang disandikan. Untuk *native* dan *mobile apps*, ada nilai default untuk *redirect uri* yaitu menuju ke *https://login.microsoftonline.com/common/oauth2/nativeclient*.
scope	*required*	Diisi dengan hak akses pada *Microsoft Graph* yang akan diberikan pada pengguna aplikasi.
response_mode	*recommended*	Nilai yang akan menentukan metode kembalian *token* yang akan dihasilkan kembali ke aplikasi. Terdapat 2 nilai yaitu "*query*" dan "*form_post*".
state	*recommended*	Nilai yang dipasang untuk mencegah serangan *cross-site request*. Parameter ini juga bisa digunakan untuk menyandikan informasi tentang keadaan pengguna di aplikasi sebelum melakukan permintaan otentikasi.

Setelah melakukan request untuk mendapatkan kode otentikasi, maka akan ditujukan ke halaman yang seperti ini.

Bentuk dari respon otentikasi seperti:

```
GET https://localhost/myapp/?
code=M0ab92efe-b6fd-f08-87dc-2c6500a7f84d
&state=12345
```

Parameter	**Deskripsi**
code	Kode otorisasi yang diminta aplikasi. Dengan kode otorisasi ini, aplikasi bisa meminta *access token* untuk mendapatkan data. Kode otorisasi memiliki umur yang singkat yaitu akan *expired* setelah 10 menit.
state	Nilai yang akan bernilai sama persis seperti saat melakukan *request* jika saat melakukan *request* memasukkan parameter "*state*" juga.

Setelah mendapat *authorize code*, maka otorisasi dilanjutkan dengan meminta kepada *Azure AD* 2.0 **/token** *endpoint* untuk mendapatkan *access token* untuk aplikasi tersebut. Aplikasi yang ingin melakukan *request* untuk mendapatkan *access token* akan memakai kode otorisasi dan juga mengirimkan *request* dengan cara "*post*". Berikut contoh format untuk melakukan *request* kepada *endpoint /token*.

```
POST /common/oauth2/v2.0/token HTTP/1.1
Host: https://login.microsoftonline.com
Content-Type: application/x-www-form-urlencoded

client_id=6731de76-14a6-49ae-97bc-6eba6914391e
&scope=user.read%20mail.read
&code=OAAABAAAAiL9Kn2Z27UubvWFPbm0gLWQJVzCTE9UkP3pSx1aXxUjq3n8b2
JRLk4OxVXr...
&redirect_uri=http%3A%2F%2Flocalhost%2Fmyapp%2F
&grant_type=authorization_code
&client_secret=JqQX2PNo9bpM0uEihUPzyrh // NOTE: Only required for web apps
```

Parameter		**Deskripsi**
tenant	*required*	Nilai yang dipakai untuk mengontrol siapa yang bisa mengakses masuk ke aplikasi. Nilainya bisa diisi dengan akun *Microsoft*.
client_id	*required*	Diisi dengan *Application ID* yang diberikan oleh portal registrasi (*apps.dev.microsoft.com*).
grant_type	*required*	Diisi dengan nilai "*authorization_code*" untuk menjalankan kode otorisasi.
scope	*required*	Diisi dengan hak akses pada *Microsoft Graph* yang akan diberikan pada pengguna aplikasi.
code	*required*	Kode otorisasi yang didapat saat melakukan request pertama.
redirect_uri	*required*	*redirect_uri* yang sama yang digunakan untuk mendapatkan kode otorisasi.
client_secret	*required for web apps*	*App secret* yang didapat saat membuat dan mendaftarkan aplikasi di portal registrasi. *Client_secret* ini hanya berlaku untuk *web apps* dan juga *web APIs* karena di *native app* tidak memungkinkan untuk menyimpan *client_secret* di gawai masing-masing.

Contoh respon jika request berhasil diproses adalah seperti berikut:

```
{
"token_type": "Bearer",
"scope": "user.read%20Fmail.read",
"expires_in": 3600,
"access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6Ik5HVEZ2ZEstZnl0
aEV1Q...",
"refresh_token": "AwABAAAAvPM1KaPlrEqdFSBzjqfTGAMxZGUTdM0t4B4..."
}
```

Setelah mendapatkan *access token*, barulah akses untuk mendapatkan data-data dari *Outlook.com Calendar* bisa diakses dengan menggunakan *Outlook.com Calendar API*. Pentingnya langkah sebelum ini dikarenakan untuk melakukan *request* kepada *Outlook.com Calendar API* diperlukan parameter *access token* saat hendak melakukan *request*. Adapun list dari *request* yang akan dipakai dari *Outlook.com Calendar API* adalah:

Request	endpoint
List events	GET /me/events GET /users/{id | userPrincipalName}/events GET /me/calendar/events GET /users/{id | userPrincipalName}/calendar/events GET /me/calendars/{id}/events GET /users/{id | userPrincipalName}/calendars/{id}/events GET /me/calendargroup/calendars/{id}/events GET /users/{id | userPrincipalName}/calendargroup/calendars/{id}/events GET /me/calendargroups/{id}/calendars/{id}/events GET /users/{id | userPrincipalName}/calendargroups/{id}/calendars/{id}/events

Request di atas akan meminta event yang ada di akun pengguna. Field {id} di atas diisi oleh id dari pengguna. Sedangkan field {userPrincipalName} bisa diisi dengan nama pengguna.

Listing 2.1: Contoh respon dari request List events diatas

```
HTTP/1.1 200 OK
Content-type: application/json
Preference-Applied: outlook.timezone="Pacific␣Standard␣Time"
Content-length: 1932

{
    "@odata.context":"https://graph.microsoft.com/v1.0/$metadata#
␣␣␣␣users('cd209b0b-3f83-4c35-82d2-d88a61820480')/events(subject,body,bodyPrev
    "value":[
        {
            "@odata.etag":"W/\"ZlnW4RIAV06KYYwlrfNZvQAAKGWwbw==\"",
            "id":"AAMkAGIAAAoZDOFAAA=",
            "subject":"Orientation␣",
            "bodyPreview":"Dana,␣this␣is␣the␣time␣you␣selected␣for␣our
␣␣␣␣␣␣␣␣␣␣␣␣␣␣orientation.␣Please␣bring␣the␣notes␣I␣sent␣you.",
            "body":{
                "contentType":"html",
                "content":"<html><head></head><body><p>Dana,␣this␣is␣the
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣time␣you␣selected␣for␣our␣orientation.␣Please␣bring␣the
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣notes␣I␣sent␣you.</p></body></html>"
            },
            "start":{
                "dateTime":"2017-04-21T10:00:00.0000000",
                "timeZone":"Pacific␣Standard␣Time"
            },
            "end":{
                "dateTime":"2017-04-21T12:00:00.0000000",
                "timeZone":"Pacific␣Standard␣Time"
            },
            "location": {
                "displayName": "Assembly␣Hall",
                "locationType": "default",
```

```
                "uniqueId": "Assembly Hall",
                "uniqueIdType": "private"
            },
            "locations": [
                {
                    "displayName": "Assembly Hall",
                    "locationType": "default",
                    "uniqueIdType": "unknown"
                }
            ],
            "attendees":[
                {
                    "type":"required",
                    "status":{
                        "response":"none",
                        "time":"0001-01-01T00:00:00Z"
                    },
                    "emailAddress":{
                        "name":"Samantha Booth",
                        "address":"samanthab@a830edad905084922E17020313.
onmicrosoft.com"
                    }
                },
                {
                    "type":"required",
                    "status":{
                        "response":"none",
                        "time":"0001-01-01T00:00:00Z"
                    },
                    "emailAddress":{
                        "name":"Dana Swope",
                        "address":"danas@a830edad905084922E17020313.
onmicrosoft.com"
                    }
                }
            ],
            "organizer":{
                "emailAddress":{
                    "name":"Samantha Booth",
                    "address":"samanthab@a830edad905084922E17020313.
onmicrosoft.com"
                }
            }
        }
    ]
}
```

2.3 *Slack API*

Slack API adalah webservice yang akan digunakan untuk menghubungkan data yang sudah di dapat dari Outlook.com Calendar ke aplikasi Slack. Disini akan dipakai API yang berfungsi untuk

mengubah status dari pengguna Slack.

2.4 *Node.js*

2.5 Cron

LAMPIRAN A

KODE PROGRAM

Listing A.1: MyCode.c

```c

// This does not make algorithmic sense,
// but it shows off significant programming characters.

#include<stdio.h>

void myFunction( int input, float* output ) {
    switch ( array[i] ) {
        case 1: // This is silly code
            if ( a >= 0 || b <= 3 && c != x )
                *output += 0.005 + 20050;
            char = 'g';
            b = 2^n + ~right_size - leftSize * MAX_SIZE;
            c = (--aaa + &daa) / (bbb++ - ccc % 2 );
            strcpy(a,"hello $@?");
    }
    count = ~mask | 0x00FF00AA;
}

// Fonts for Displaying Program Code in LATEX
// Adrian P. Robson, nepsweb.co.uk
// 8 October 2012
// http://nepsweb.co.uk/docs/progfonts.pdf
```

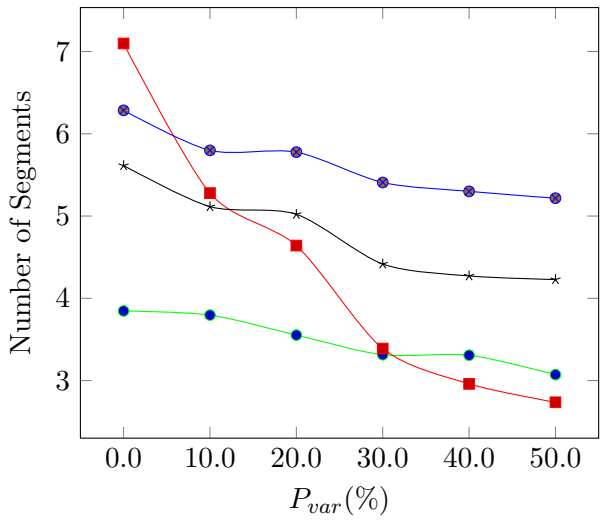
Listing A.2: MyCode.java

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;

//class for set of vertices close to furthest edge
public class MyFurSet {
    protected int id;                                   //id of the set
    protected MyEdge FurthestEdge;                      //the furthest edge
    protected HashSet<MyVertex> set;                    //set of vertices close to furthest edge
    protected ArrayList<ArrayList<Integer>> ordered;    //list of all vertices in the set for each trajectory
    protected ArrayList<Integer> closeID;               //store the ID of all vertices
    protected ArrayList<Double> closeDist;              //store the distance of all vertices
    protected int totaltrj;                             //total trajectories in the set

    /*
     * Constructor
     * @param id : id of the set
     * @param totaltrj : total number of trajectories in the set
     * @param FurthestEdge : the furthest edge
     */
    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
        this.id = id;
        this.totaltrj = totaltrj;
        this.FurthestEdge = FurthestEdge;
        set = new HashSet<MyVertex>();
        ordered = new ArrayList<ArrayList<Integer>>();
        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
        closeID = new ArrayList<Integer>(totaltrj);
        closeDist = new ArrayList<Double>(totaltrj);
        for (int i = 0;i <totaltrj;i++) {
            closeID.add(-1);
            closeDist.add(Double.MAX_VALUE);
        }
    }

}
```

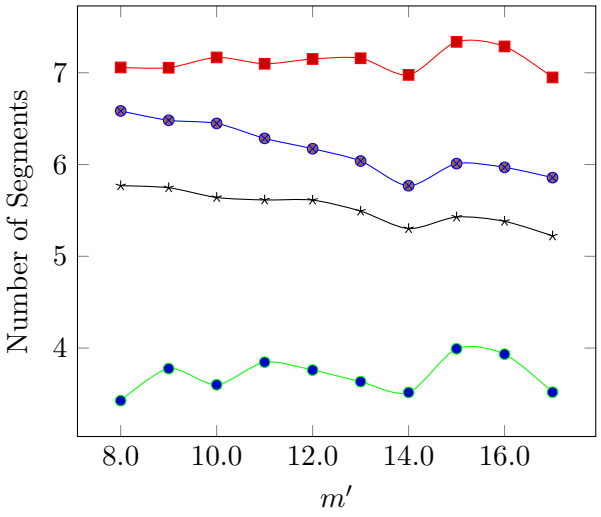
LAMPIRAN B

HASIL EKSPERIMEN

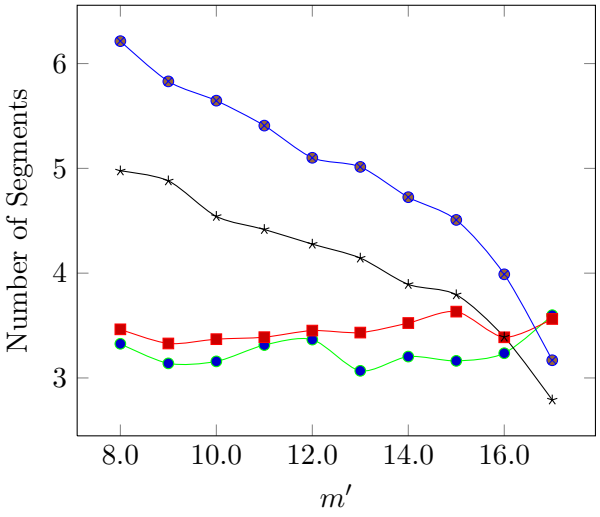
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



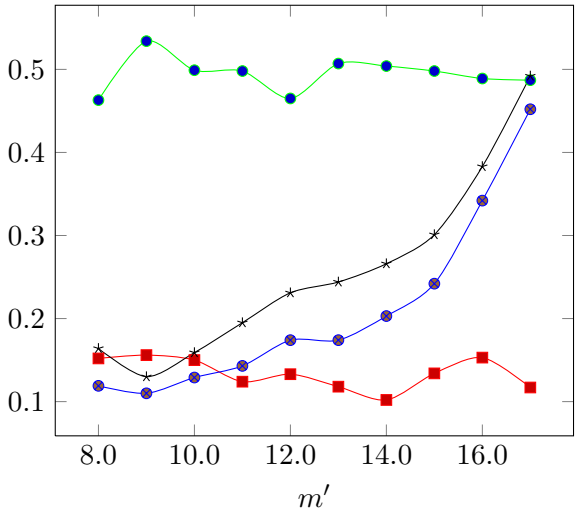
Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4