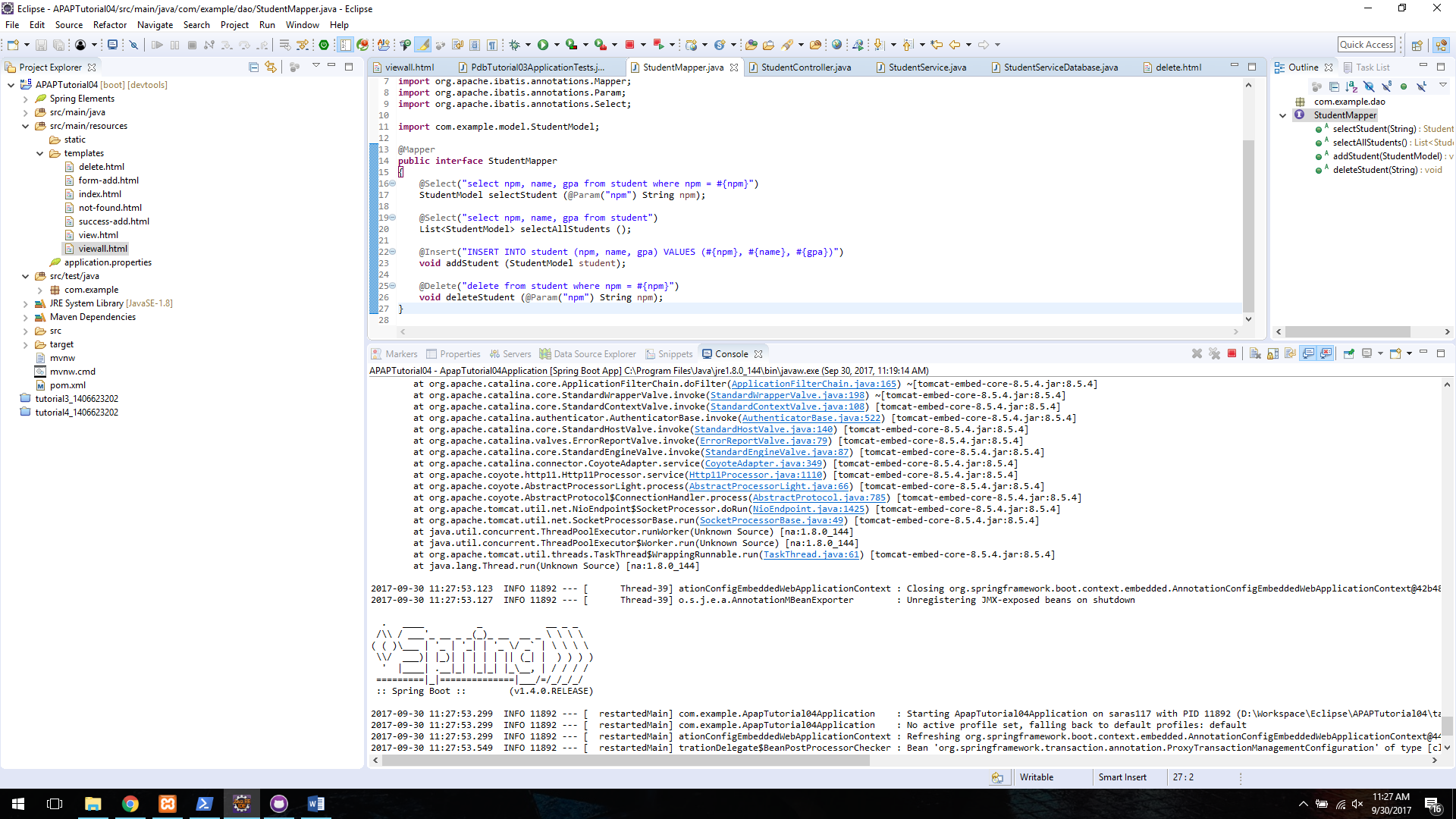
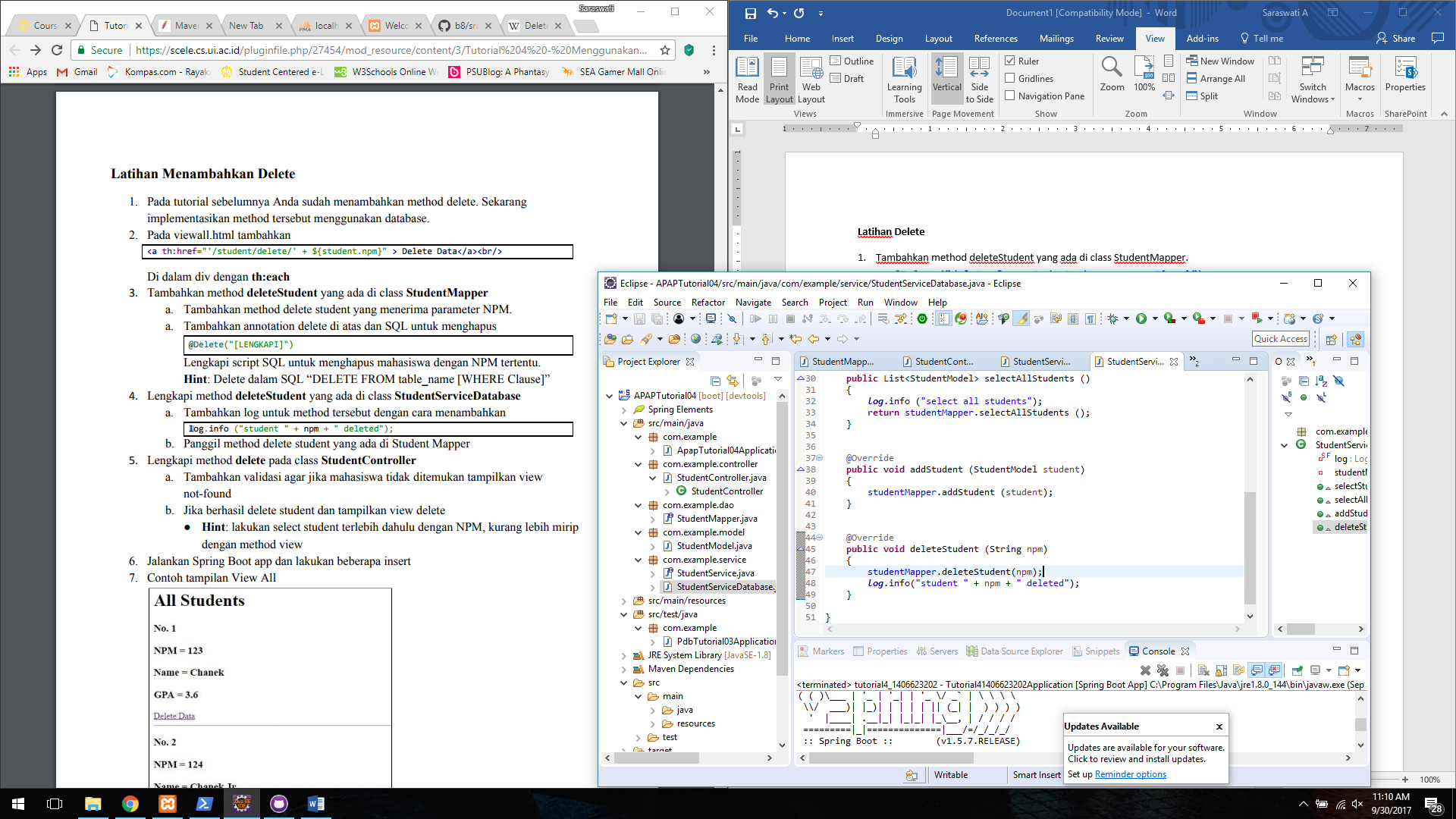
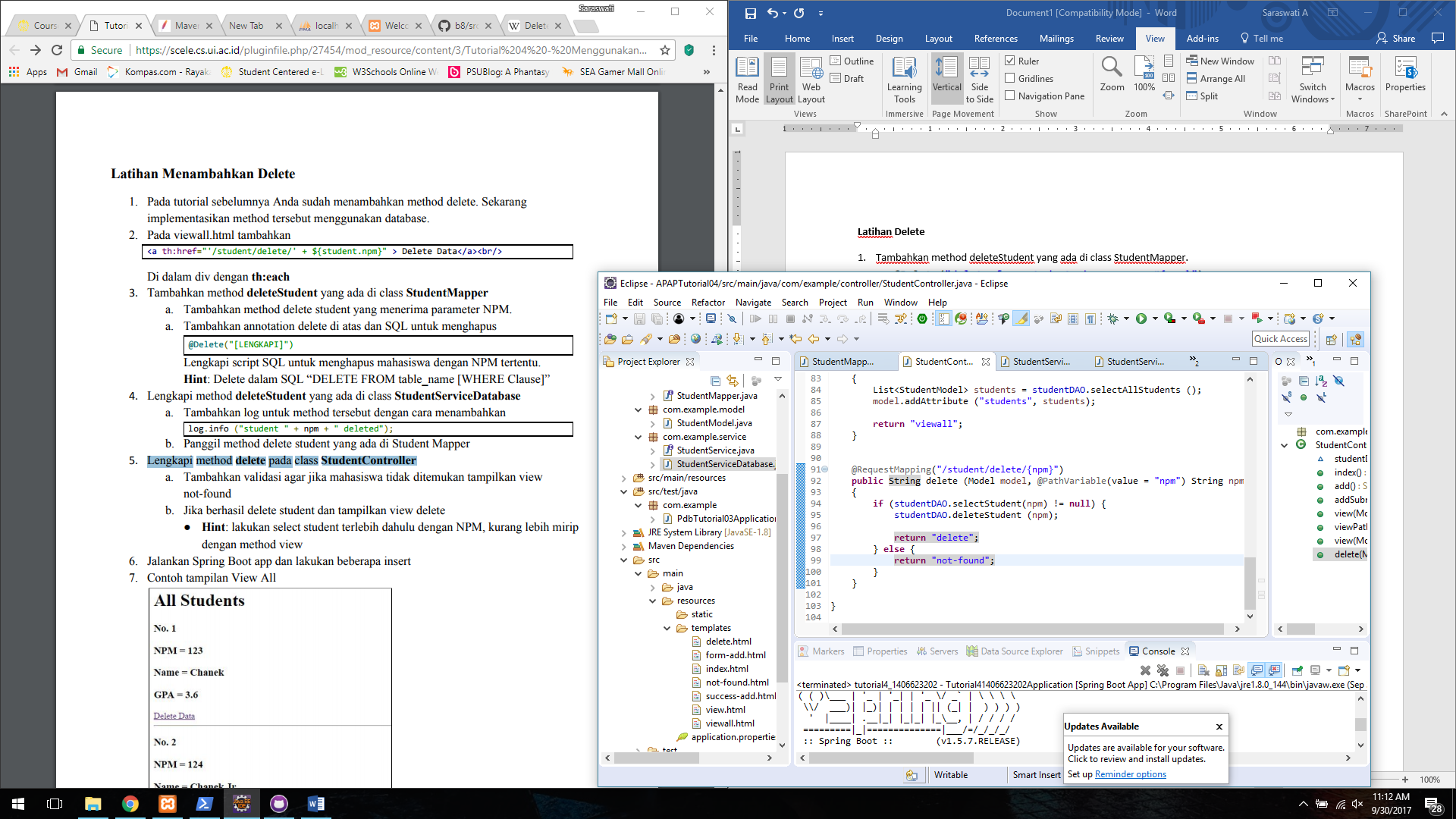
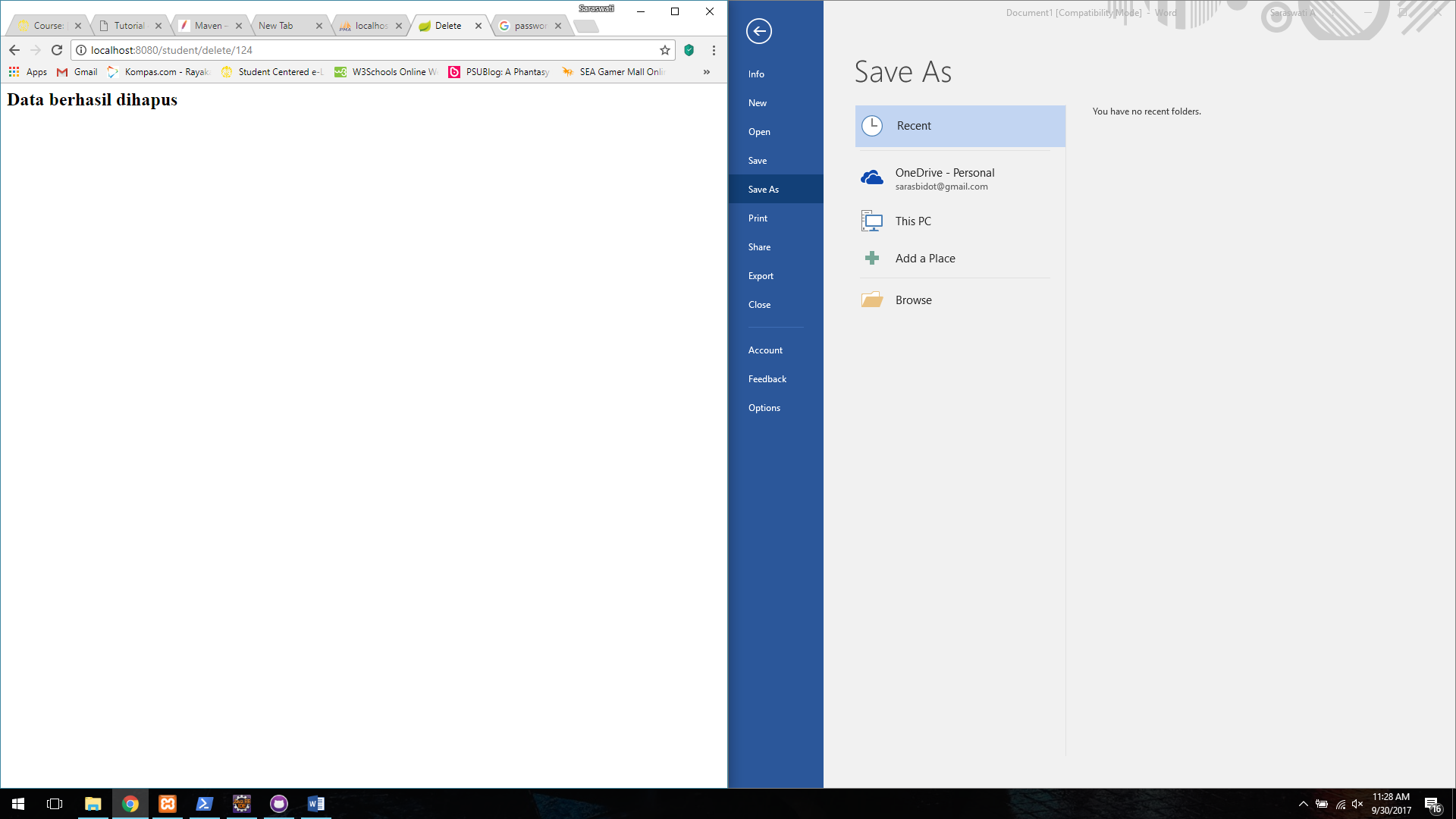
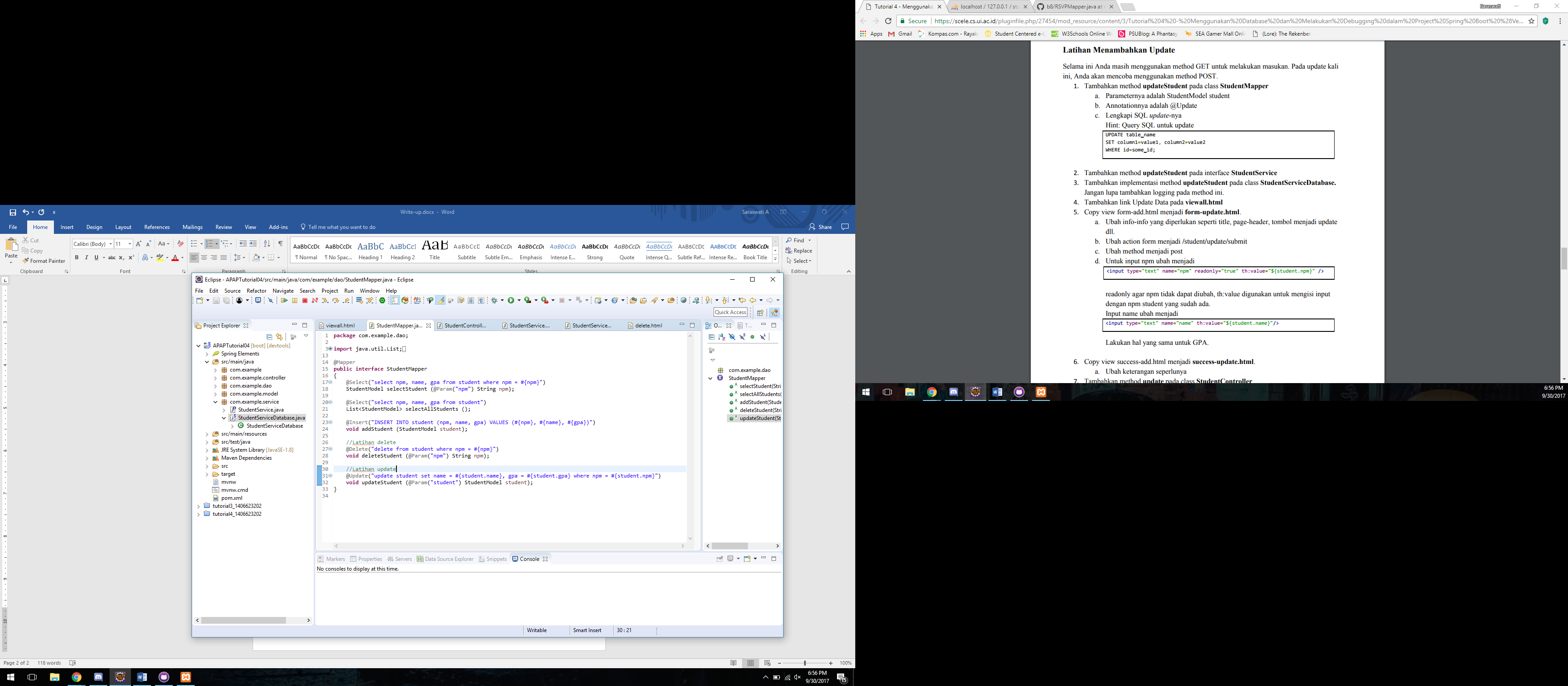
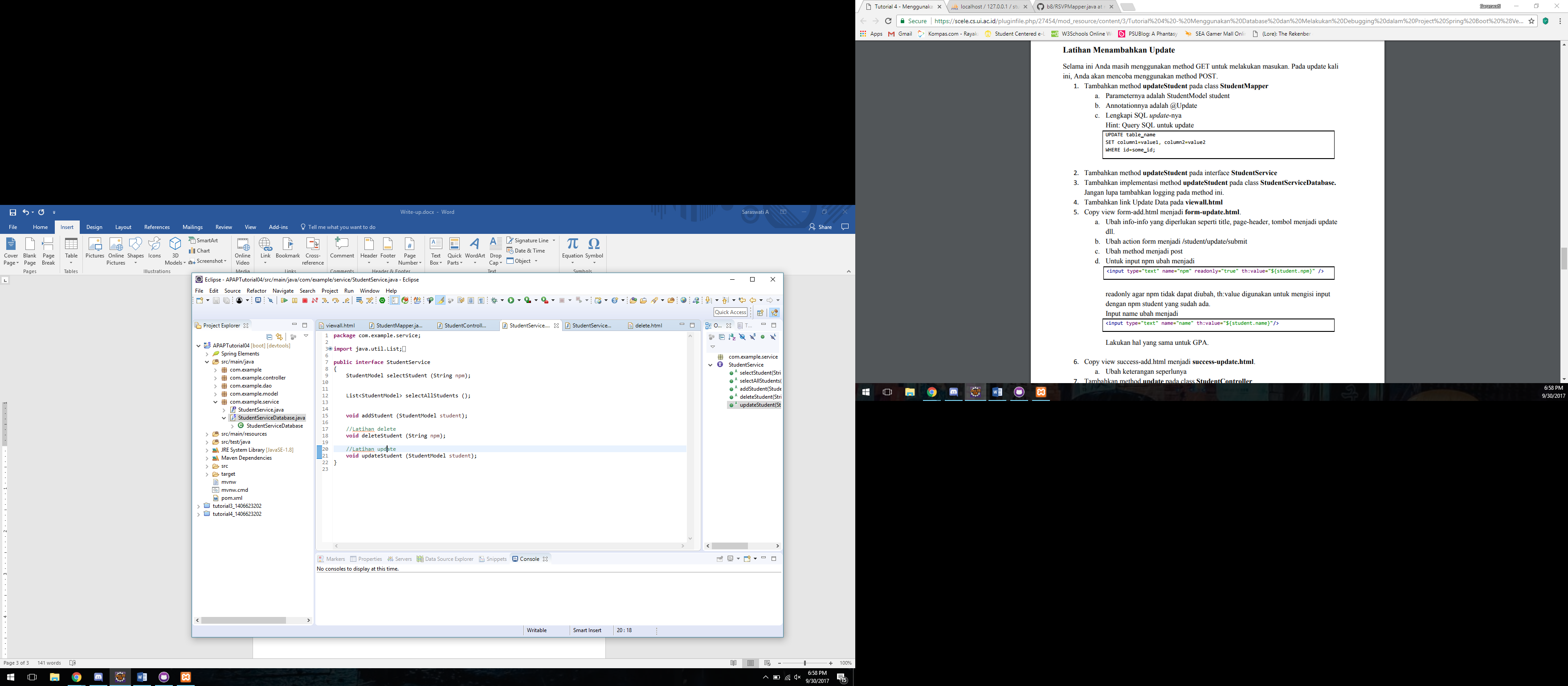
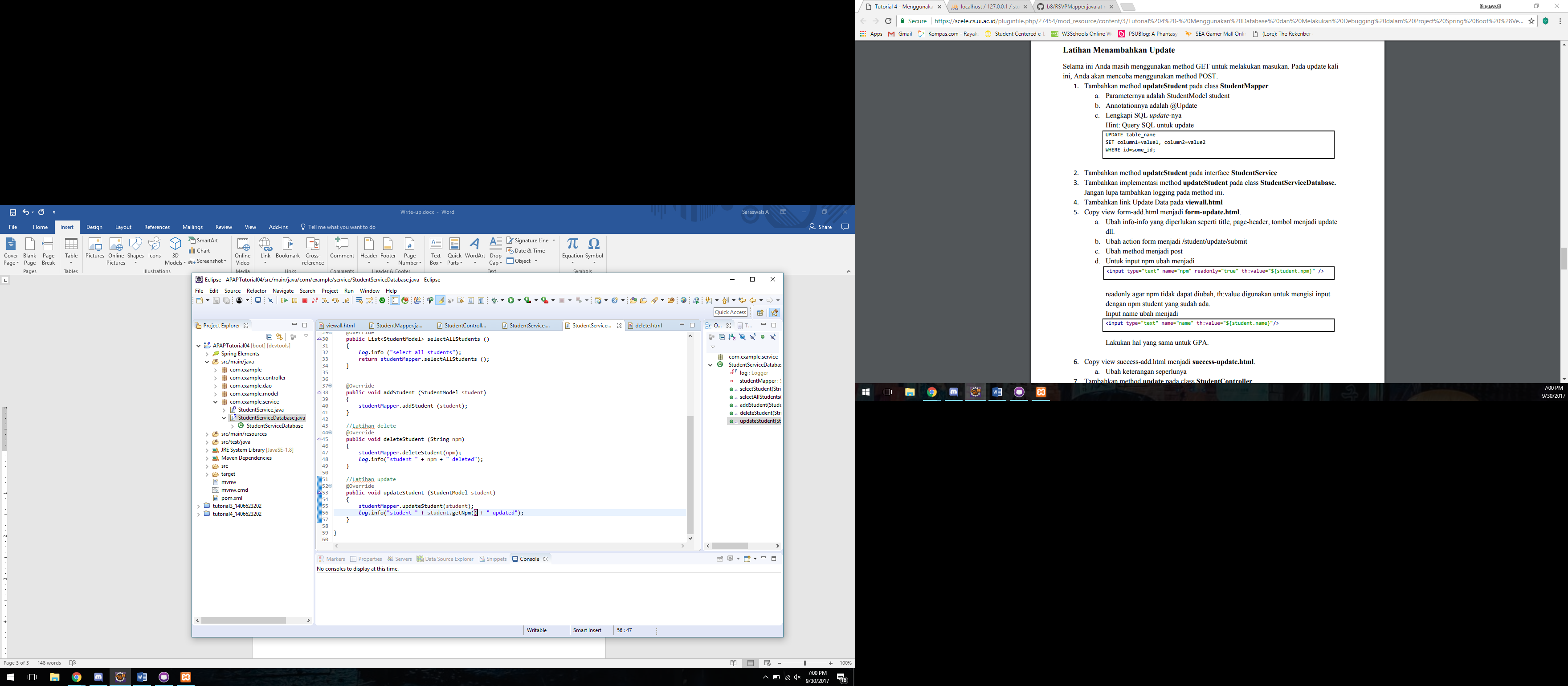
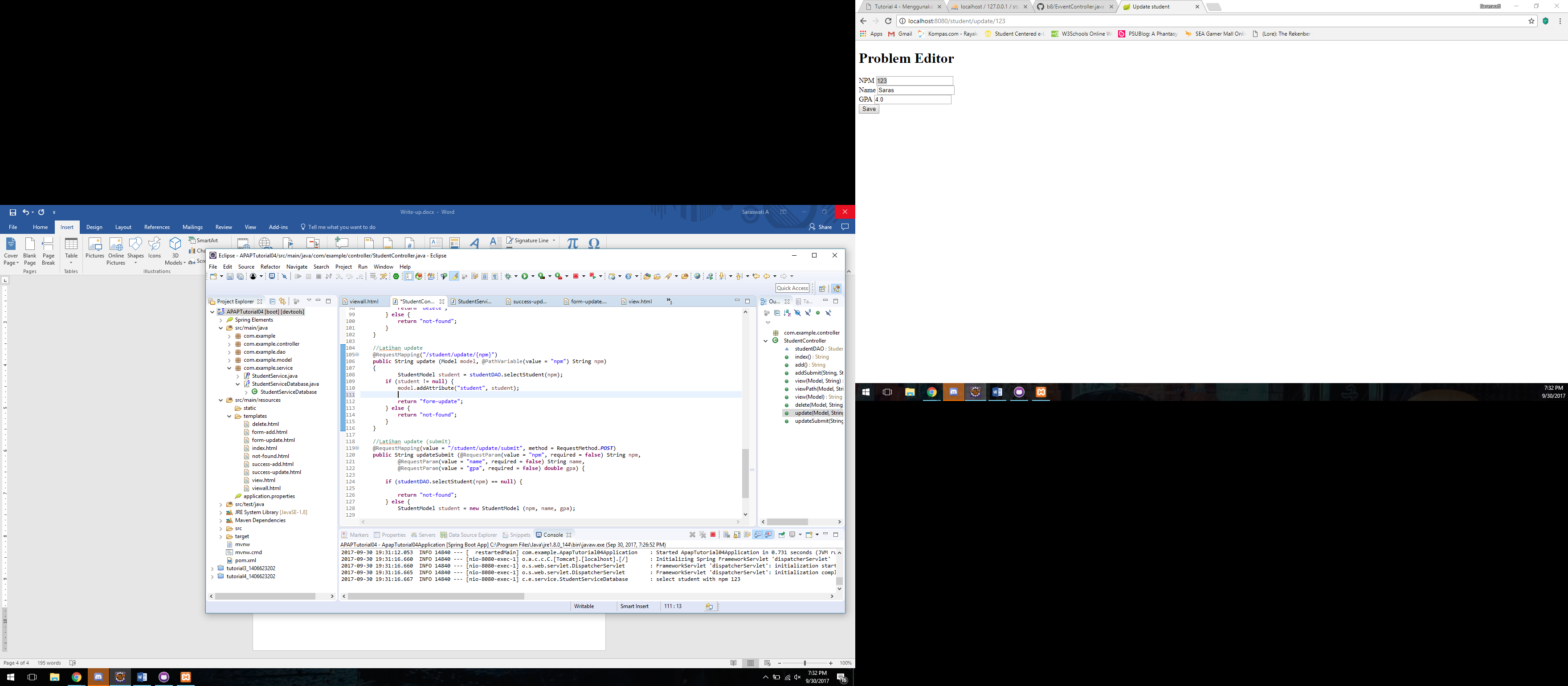
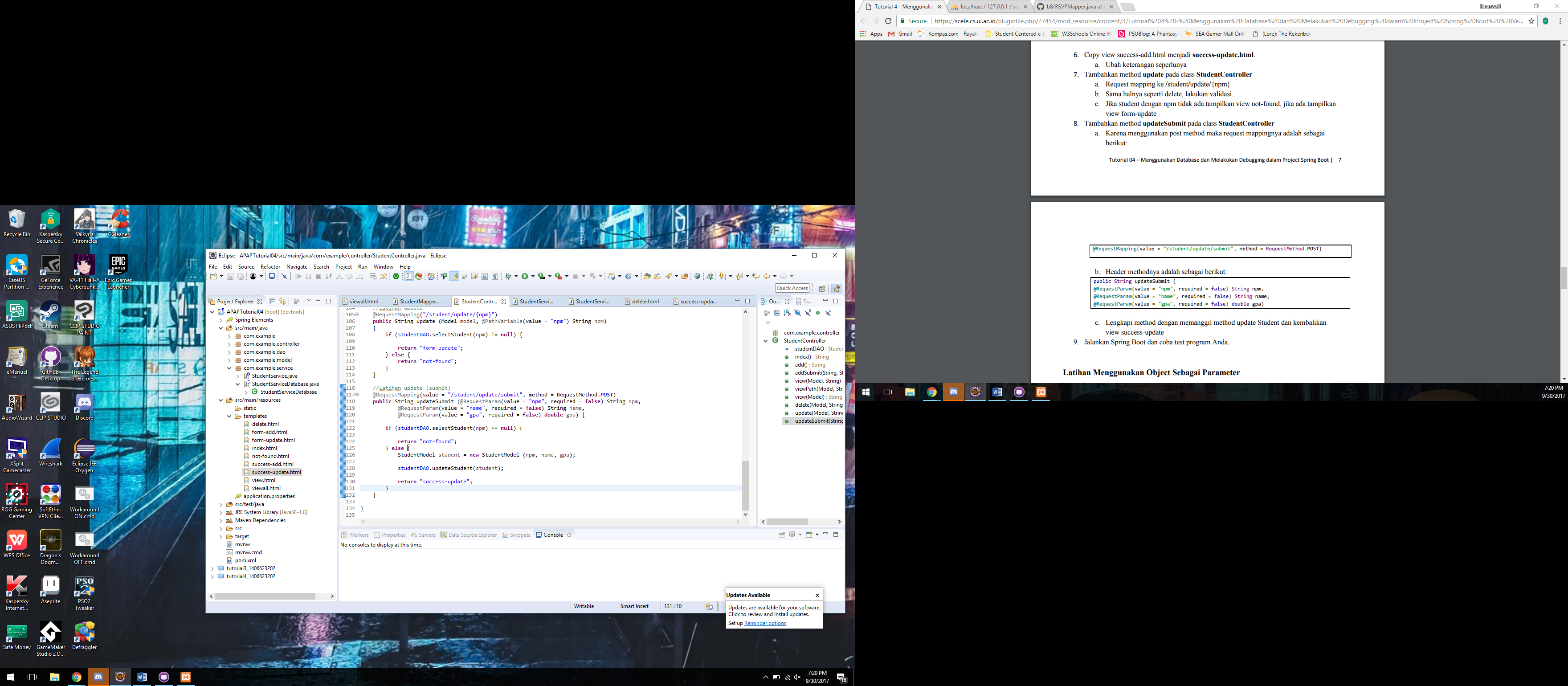
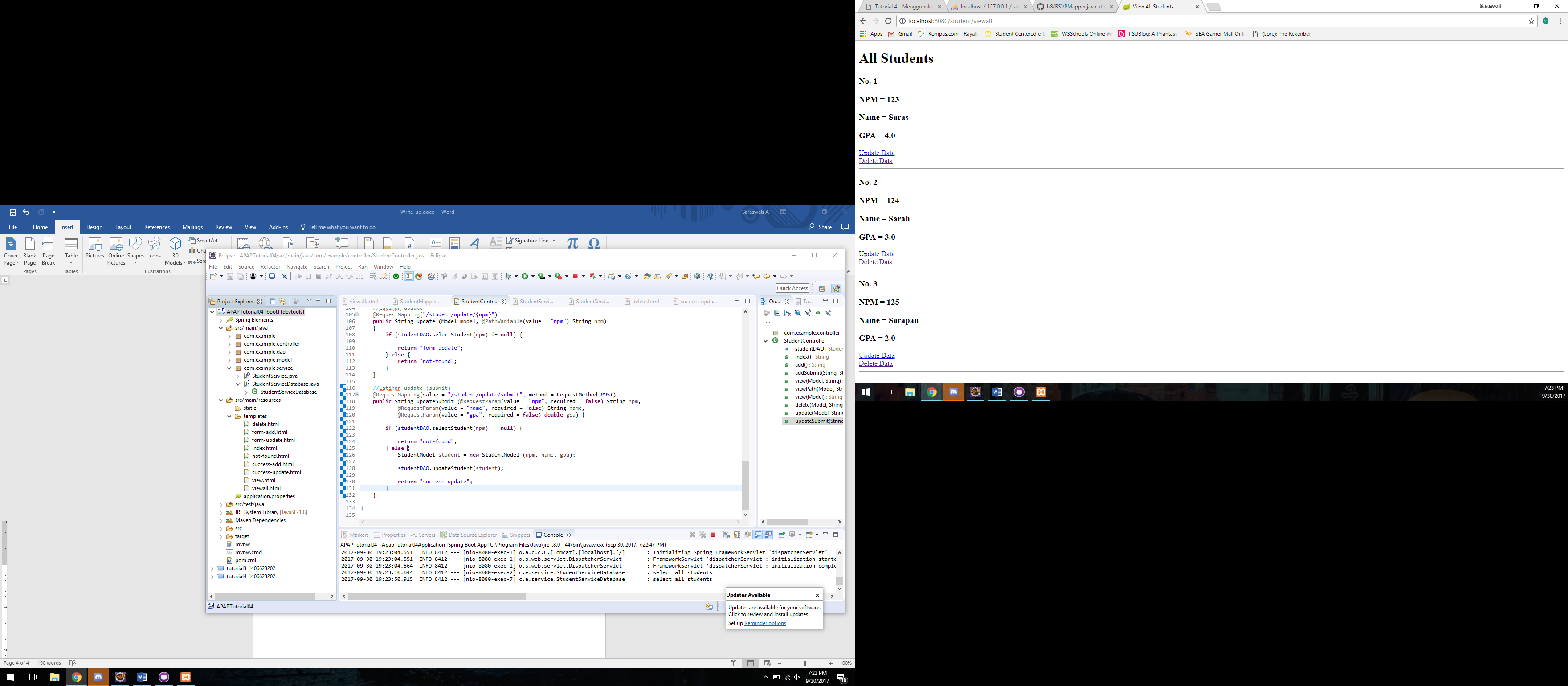
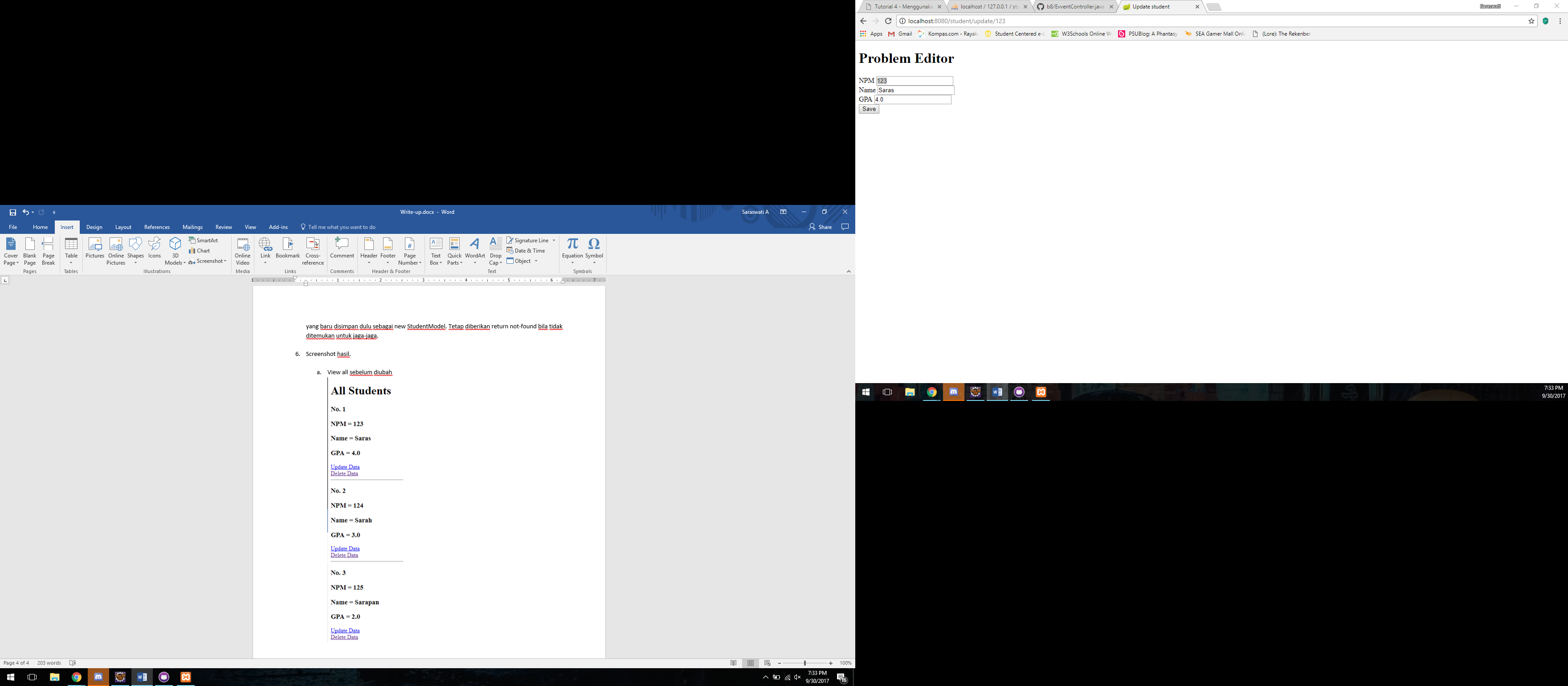
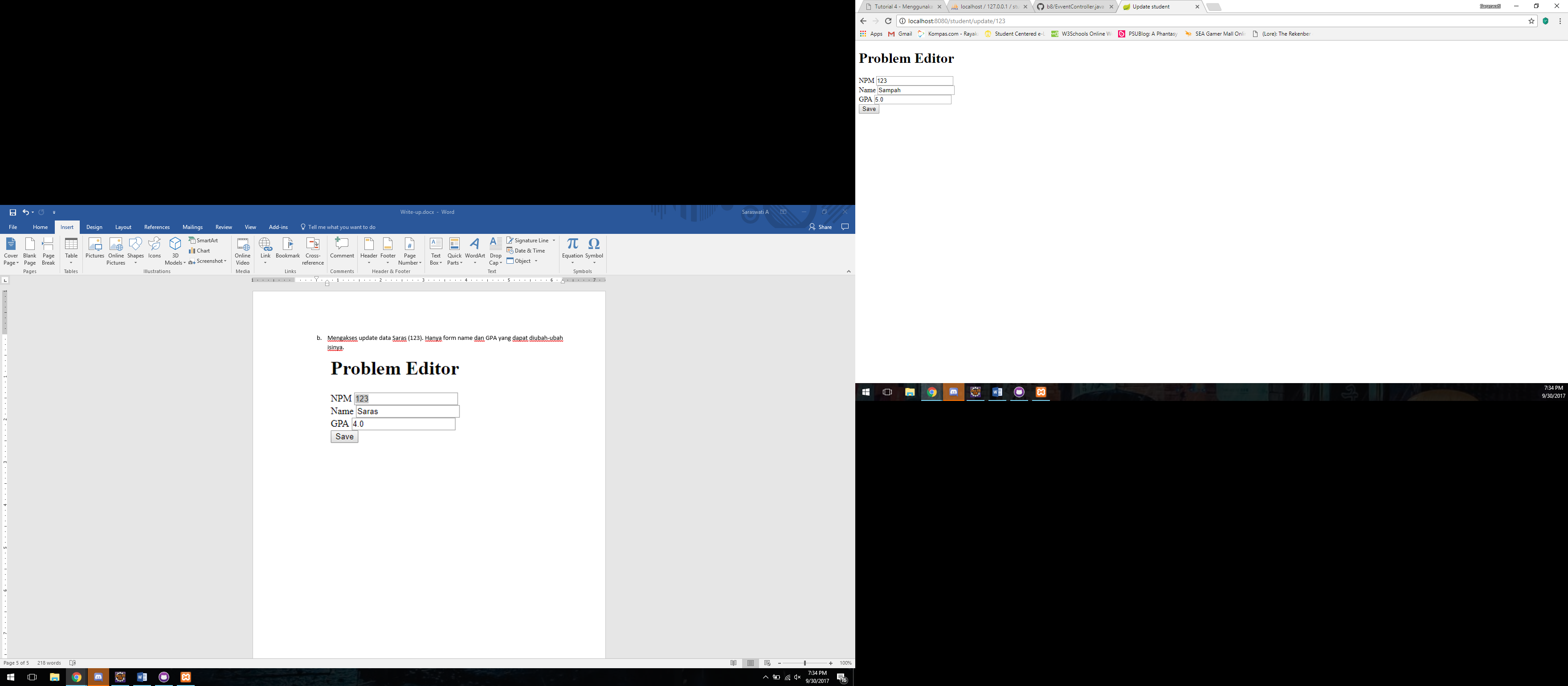
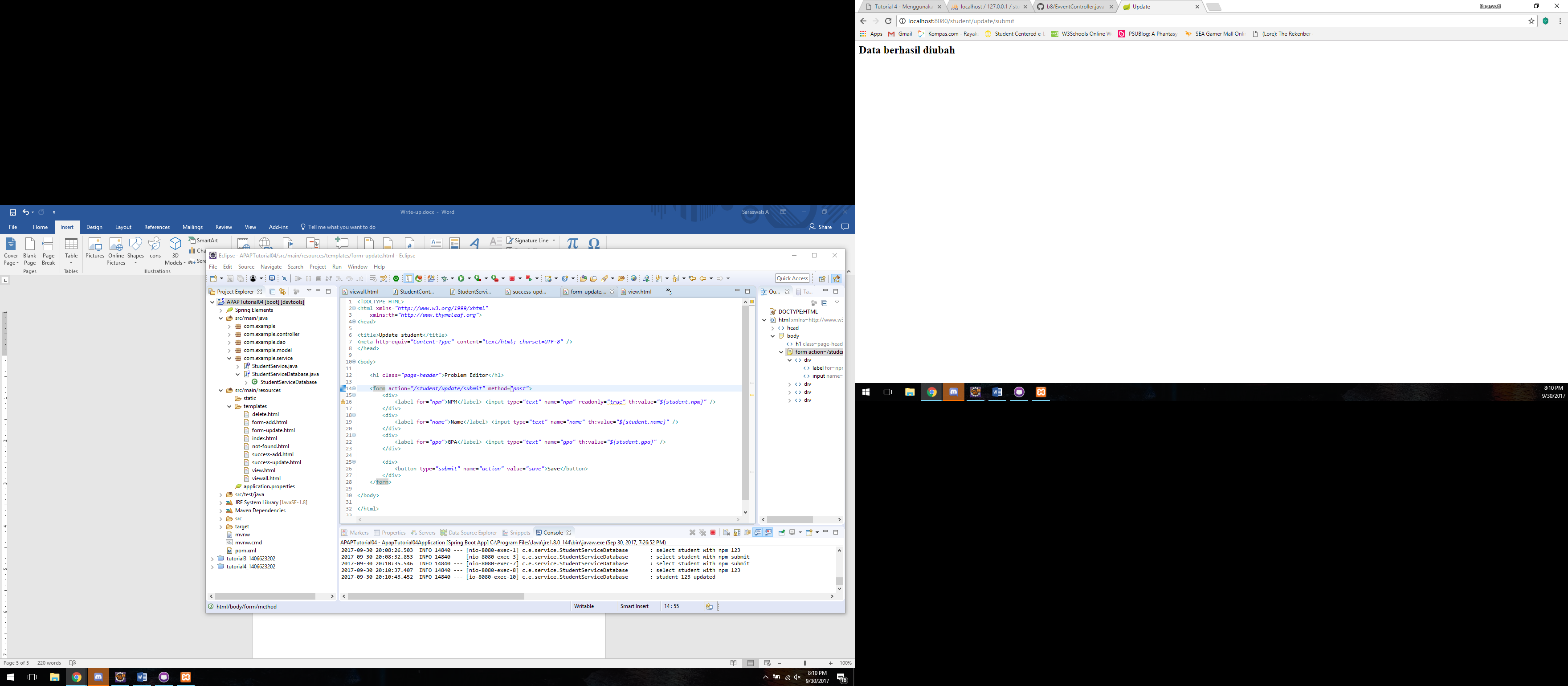
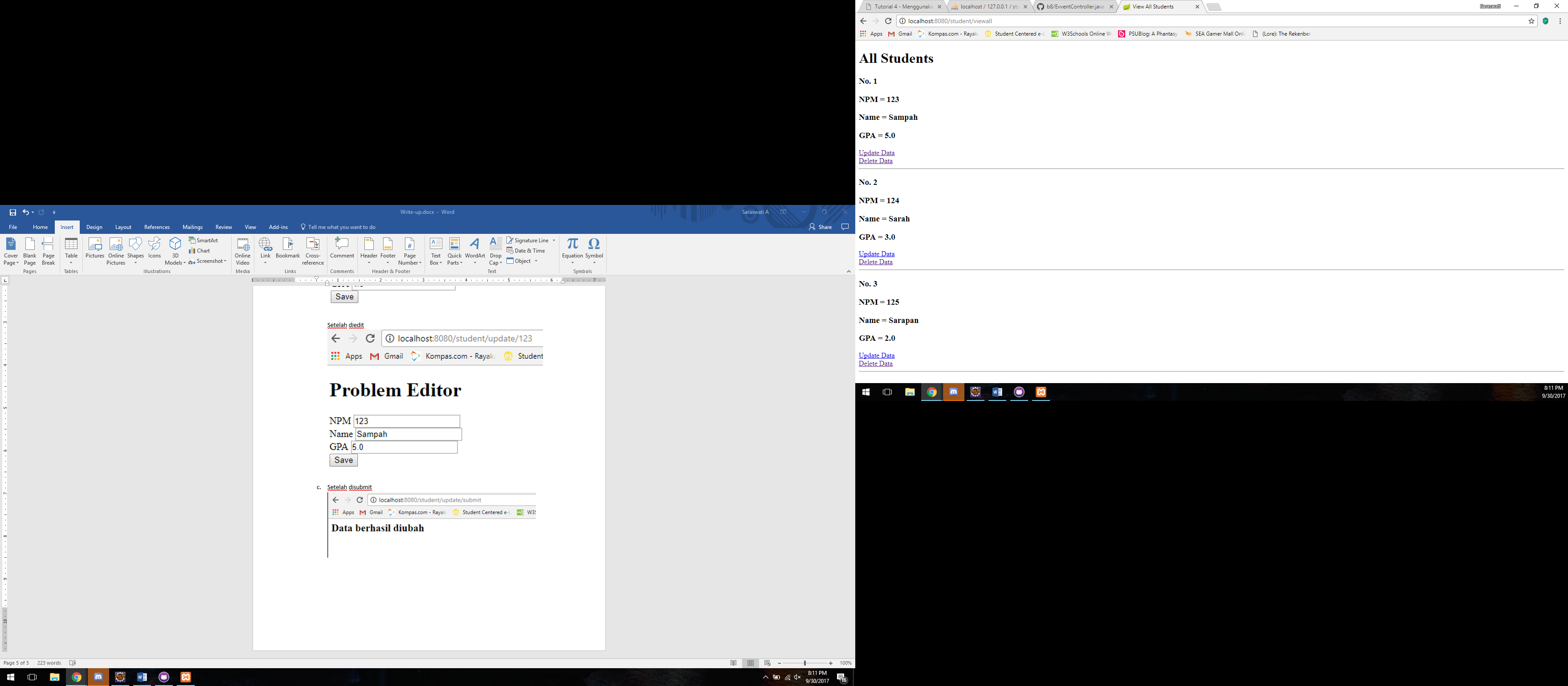
**Latihan Delete**

1. Tambahkan method deleteStudent​ ​yang ada di class StudentMapper.  
     
   Student yang akan dihapus dicari berdasarkan NPMnya, karena NPM merupakan primary key tabel student. Mirip seperti method selectStudent.
2. Lengkapi method deleteStudent​ yang ada di class StudentServiceDatabase  
     
   Method ini hanya memanggil method deleteStudent yang ada di StudentMapper dan log.info untuk mengecek proses di console Eclipse.
3. Lengkapi method delete​ pada class StudentController  
     
   Method delete di StudentController pertama mengecek dulu apakah student yang memiliki NPM tersebut ada atau tidak, jika ada maka akan lanjut ke deleteStudent. Jika tidak ada maka controller langsung mengembalikan halaman not-found.html.
4. Screenshot hasil
   1. All students  
      
   2. Menghapus data NPM 124  
      
   3. Setelah dihapus  
      

**Latihan Update**

1. Tambahkan method updateStudent pada class **StudentMapper**  
     
   Yang dapat diubah hanya nama dan gpa saja, npm digunakan untuk mencari student yang ingin diubah.
2. Tambahkan method updateStudent pada interface **StudentService**  
   
3. Impementasi method updateStudent pada class **StudentServiceDatabase**  
   
4. Tambahkan update pada class **StudentController**

  
Memasukkan attribute student agar dapat digunakan di HTML.

1. Tambahkan method updateSubmit pada class **StudentController**  
   Mirip dengan delete, tetapi karena parameter updateStudent adalah StudentModel, data-data yang baru disimpan dulu sebagai new StudentModel. Tetap diberikan return not-found bila tidak ditemukan untuk jaga-jaga.
2. Screenshot hasil.
   1. View all sebelum diubah  
      
   2. Mengakses update data Saras (123). Hanya form name dan GPA yang dapat diubah-ubah isinya.  
        
      Setelah diedit  
      
   3. Setelah disubmit  
      
   4. Tampilan di viewall  
      

**Latihan menggunakan object sebagai parameter**

1. Jika menggunakan Object sebagai parameter pada form POST, bagaimana caranya melakukan validasi input yang optional dan input yang required seperti jika menggunakan RequestParam? Apakah validasi diperlukan? Asumsikan input pada form Anda tidak menggunakan attribute required sehingga butuh validasi di backend.
2. Menurut Anda, mengapa form submit biasanya menggunakan POST method dibanding GET method? Apakah perlu penanganan berbeda di header atau body method di controller jika form di post dikirim menggunakan method berbeda?
3. Apakah mungkin satu method menerima lebih dari satu jenis request method, misalkan menerima GET sekaligus POST?