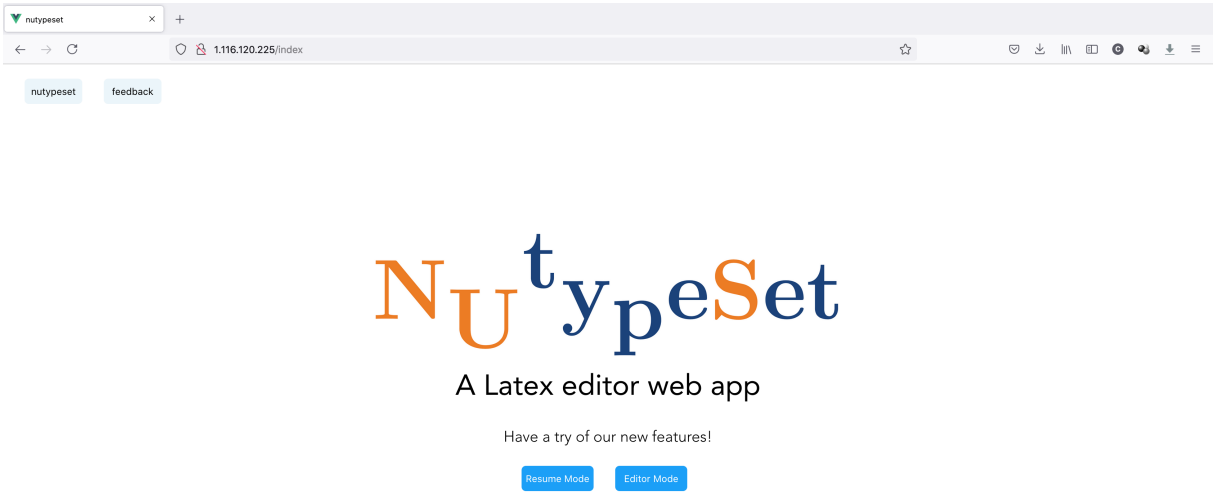
Proposal for Orbital 2021

Team Name: NUtypeSet

Proposed Level Of Achievement: Apollo 11



NUtypeSet URL: `http://1.116.120.225/`

Motivation

Do you believe you can make a **professional resume** within five minutes? You may know that **LaTex** can help make CV tidy and professional, but the complicated code would crash you and take you a lot of time to learn if you are not a professional programmer in Latex. The LaTex was made by programmer, but why not making a simpler web-site based on Latex to let it benefit more people? Nutypeset motivated to create smart **resume generator** and easy **Latex Editor** to provide convenient typeset solutions.

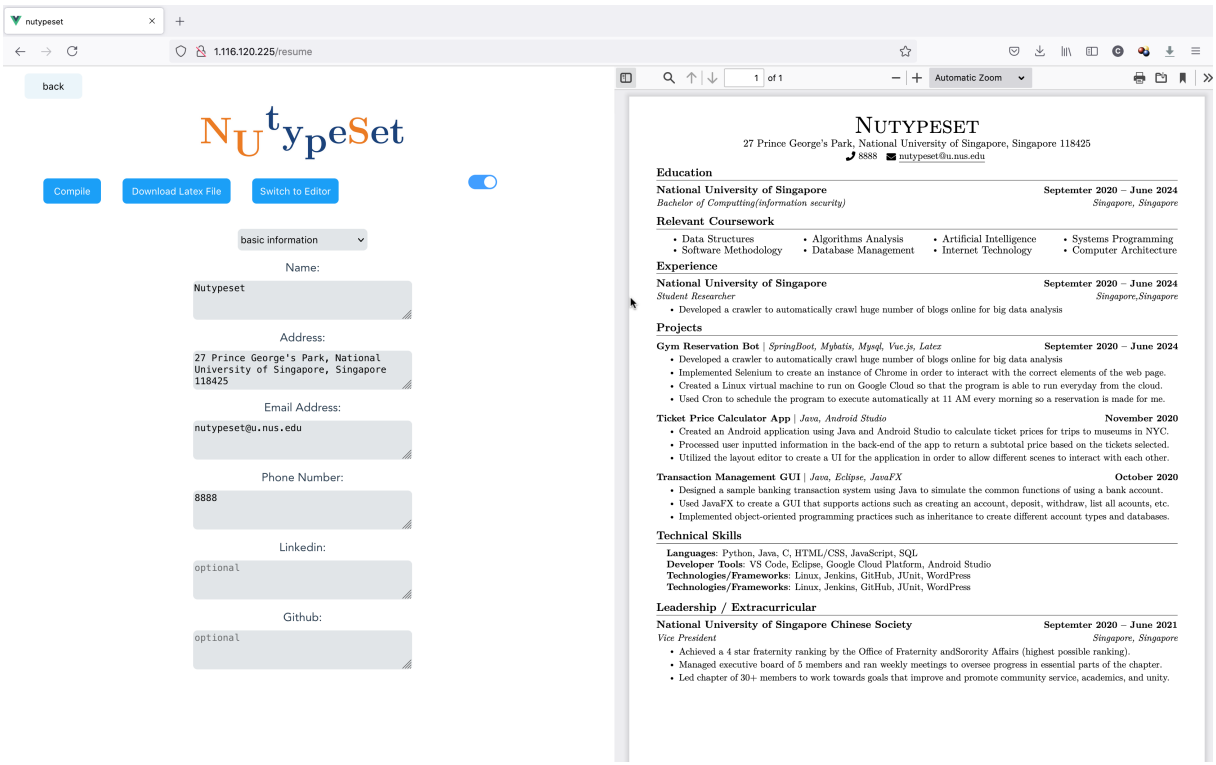
Aim

We hope to makes a convenient user-friend Latex editor with GUI mainly targeting to user with **non-programming/general** background who would like to make their **first** professional resume rapidly. There's also option for professional user who is lacked to setup environment of LaTex.

User Stories

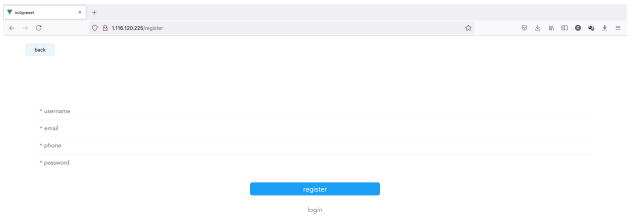
1. As a candidates who strive for making her/his resume professional but lacks typeset skill, NUtypeSet will help!
2. As a professional typeset programmer who wants to typeset a resume with Latex, NUtypeSet could save the time!
3. As an user who used other resume generator online, but found that the resume cannot be modified elsewhere, NUtypeSet provide both pdf and LaTex source code, which can be modified anywhere
4. As an user who used other resume generator online and thought the typeset on the other website are not so perfect, NUtypeSet provide not only professional template, but also LaTex easy editor for further professional modification

Features

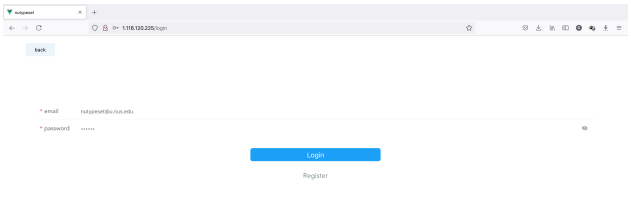


1. The website provides a web-base forum-like window, in which users can input basic component of an resume in template(e.g. basic information, education, work experience, etc.) and the server will typeset the resume with LaTex automatically(Finished) (Finished).
2. the website also provides additional latex editor for further resume editing and other purpose that needs LaTex environment.
3. Server can compile the LaTex file automatically. (Finished).
4. Generated resume can be previewed. (Finished).

5. User can download resume. (Finished).
6. User can download LaTex file. (Finished).

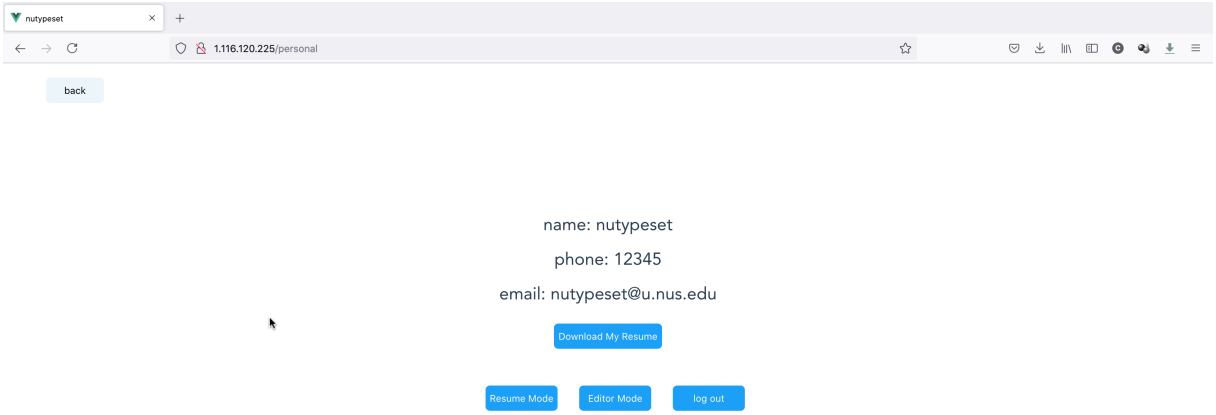


Register



Login

7. Register and login(Finished).
8. Account and password management(Finished).



9. Storage of User's resume and information(Finished).
10. Auto crawl user's information from user's Linkedin when user allows for generating resume
11. An AI for evaluating user's resume and giving suggestions.

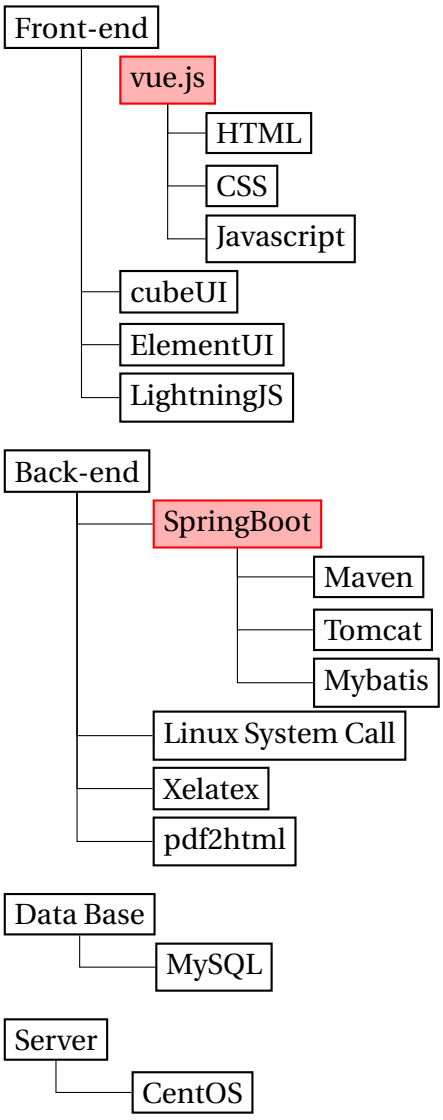
Timeline

TABLE 1 Timeline

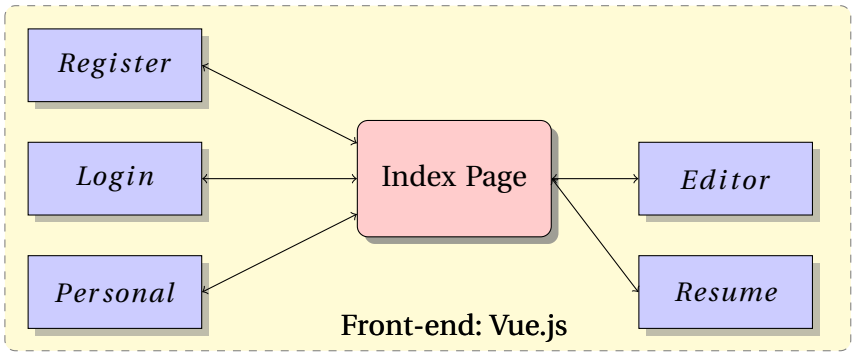
By 10 May	A clear plan for the server and language in each part
By 31 May	A WYSIWYG GUI(similar to jupyter notebook), where user can enter latex code and get the running feedback
Milestone1	
By 15 June	A web-base forum-like window that user can simply enter the basic components(e.g. basic information, education and work experience) and the text with the specified format will be generated. PDF will be available as well for user to download.
By 20 June	A forum-based resume generator that can generate both latex code and pdf file of the resume
Milestone2	
By 14 July	A mechanism for log in, user-based database, image, and a Latex editor UI
By 20 July	Home page
By 24 July	Personal page
By 20 July	Password management, penetration test and pressure resistent
Milestone3	
By 17 August	User profile refinement/Allows user to sync information from Linkedin to profile on NUtypeSet, and user could use the information select from their profile to generate different resume.
Future	AI may be introduced to resume marking and improvement/support additional language resume generation/more model of resume may be added

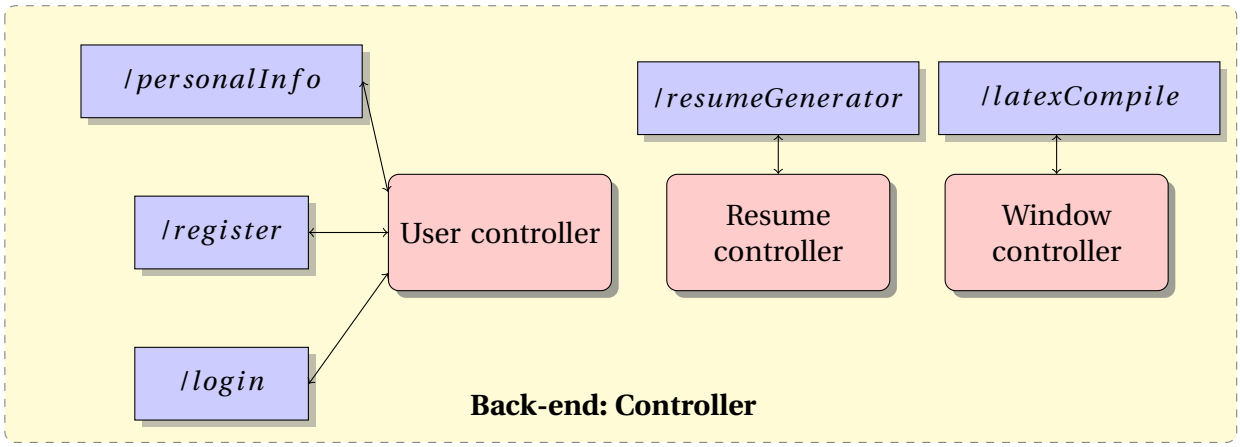
Architecture

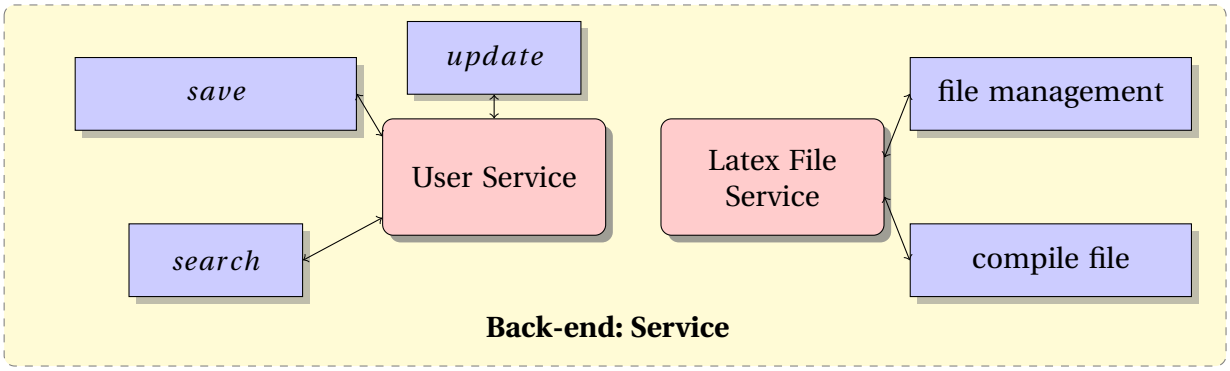
Technical Structure

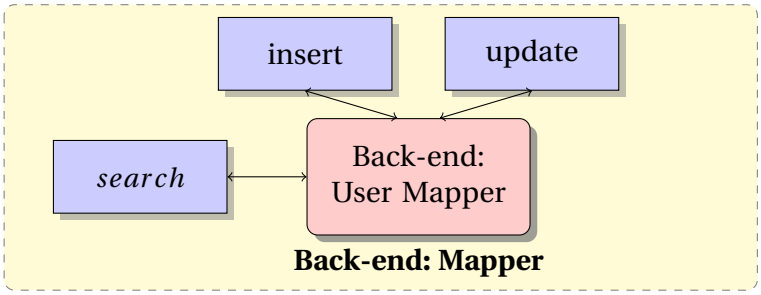


Functional Structure









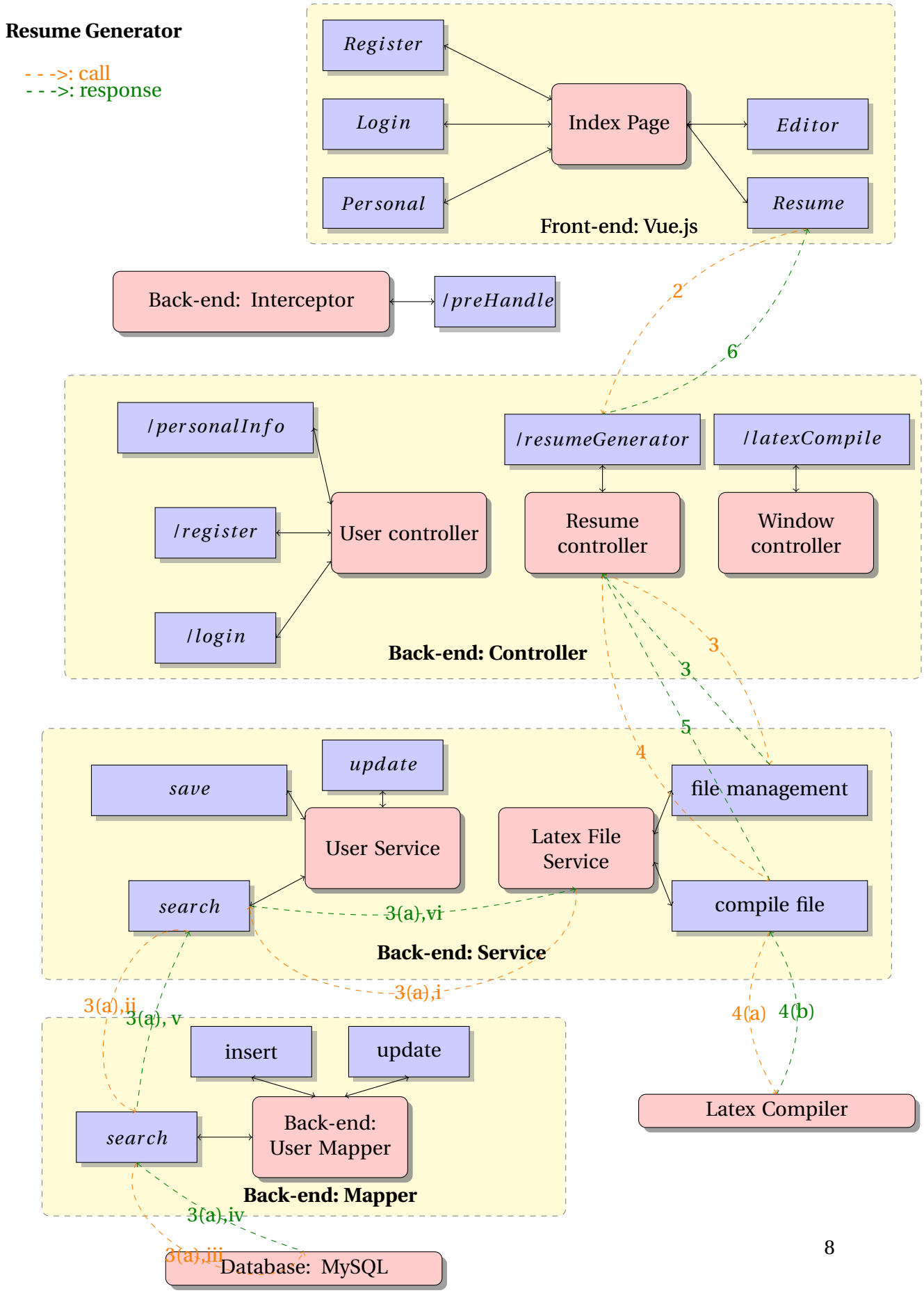
Latex Compiler

Database: MySQL

Functions

Resume Generator

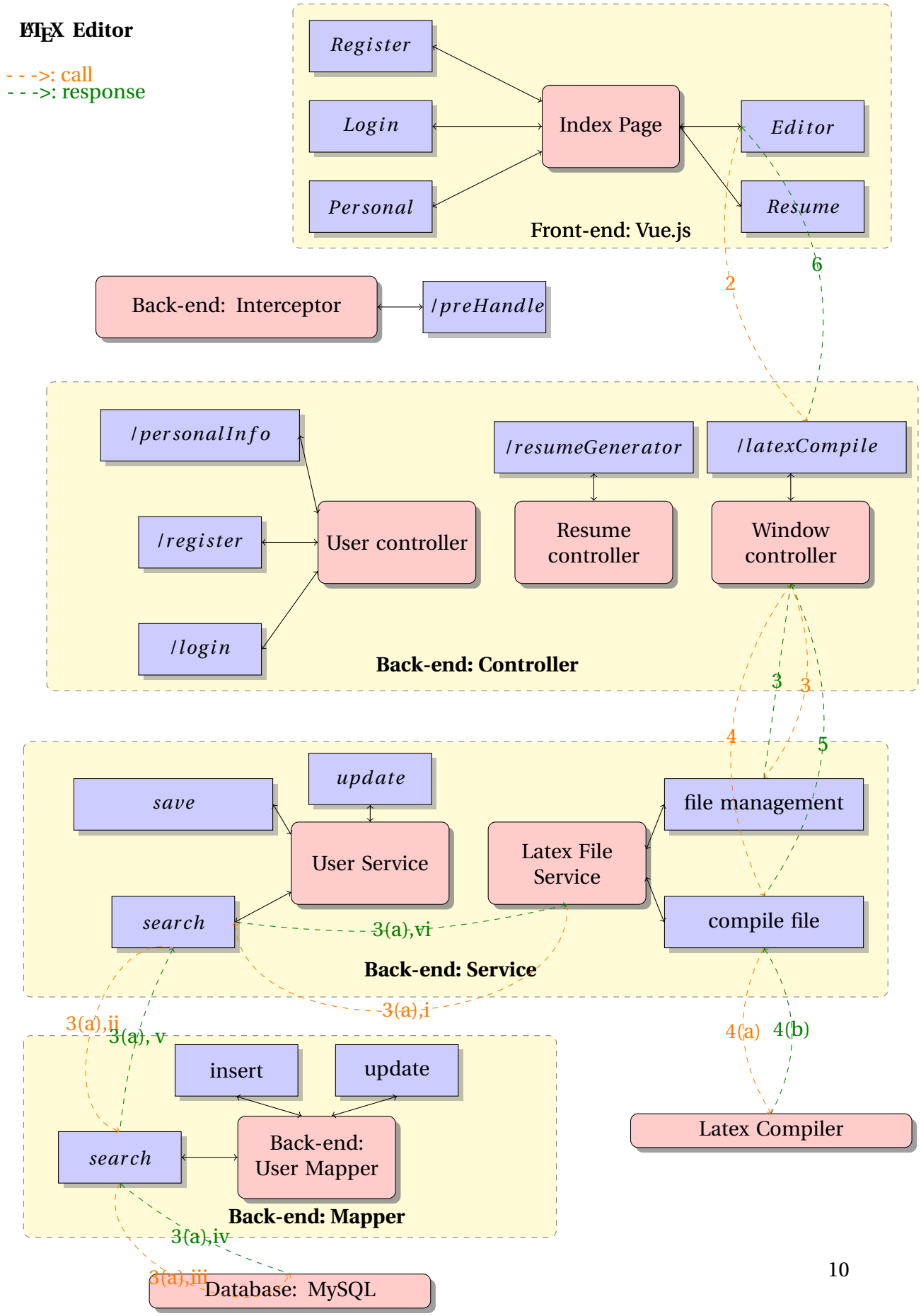
1. Resume generator page on front-end is composed of multiple component, where each component is in charge of one part of the resume, while user update the data in each input area, the data in each component will be synchronized(VueX and signal communication between child component and parent component)
2. When user click "Compile", resume page will apply API to back-end controller "/resumeGenerator", note that the data passed to the API are not generated code, but information inputted by user
3. "/resumeGenerator" will pass the information to back-end service: "Latex File Service:file management", , which will:
   (a) call the search service to valid whether the user login or not (if login, the file will be managed to certain file belongs to the user
   (b) generate the latex code and check whether there is code injection
   (c) system call to save the generated latex code and manage the file
4. Then "/resumeGenerator" will call back-end service: "Latex File Service:compile file", the "compile file" will
   (a) call Latex compiler to compile the latex file
   (b) return the address to the generated file
5. "Back-end:Service:Compile file" will generate an address to the generated file and return the address and other information to "Back-end:Controller:resumeGenerator"
6. "Back-end:Controller:resumeGenerator" will response to "Front-end:resume" with the address and some other information
7. "Front-end:resume" will update the url to the generated file and update PDF preview


Resume Generator
- - ->: call
- - ->: response
Register
Login
Personal
Index Page
Editor
Resume
Front-end: Vue.js
Back-end: Interceptor
/preHandle
2
6
/personalInfo
/register
/login
User controller
/resumeGenerator
/latexCompile
Resume controller
Window controller
Back-end: Controller
3
3
4
5
update
save
search
User Service
Latex File Service
file management
compile file
3(a),vi
Back-end: Service
3(a),i
3(a),ii
3(a), v
4(a)
4(b)
insert
update
search
Back-end: User Mapper
Back-end: Mapper
Latex Compiler
3(a),iv
3(a),iii
Database: MySQL

LaTeX Editor

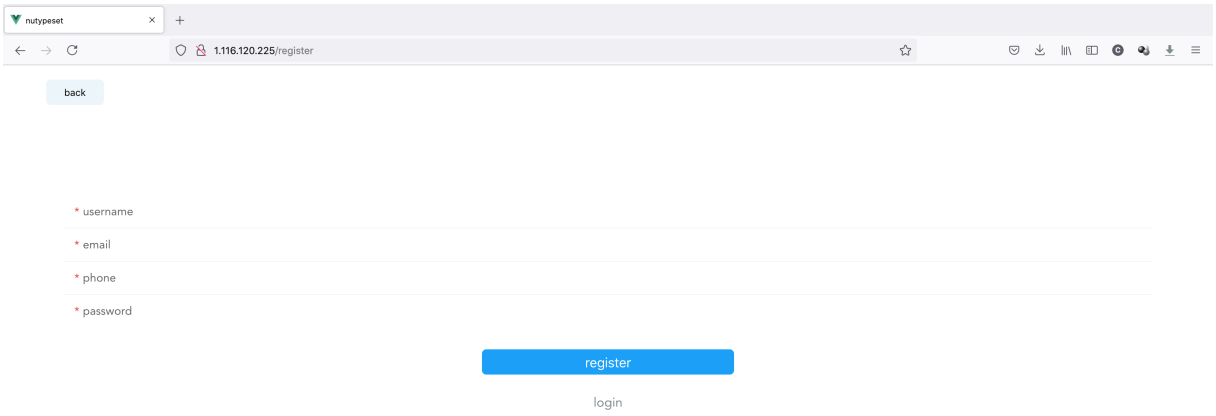
1. Front-end: CodeMirror is introduced to optimize latex editor. CodeMirror provides varies option for customization, including line number, theme and bracket matching. The source of resume will be passed to editor if user has logged in initially. When user type code in textarea, CodeMirror will read and highlight the code. At the same time, the data in textarea will be emptyed. Therefore, when the focus is blur, the code in editor will be passed back to textarea by using getValue API. The code in the editing box is synchronized with resume mode initially. User can input code in the editing box.

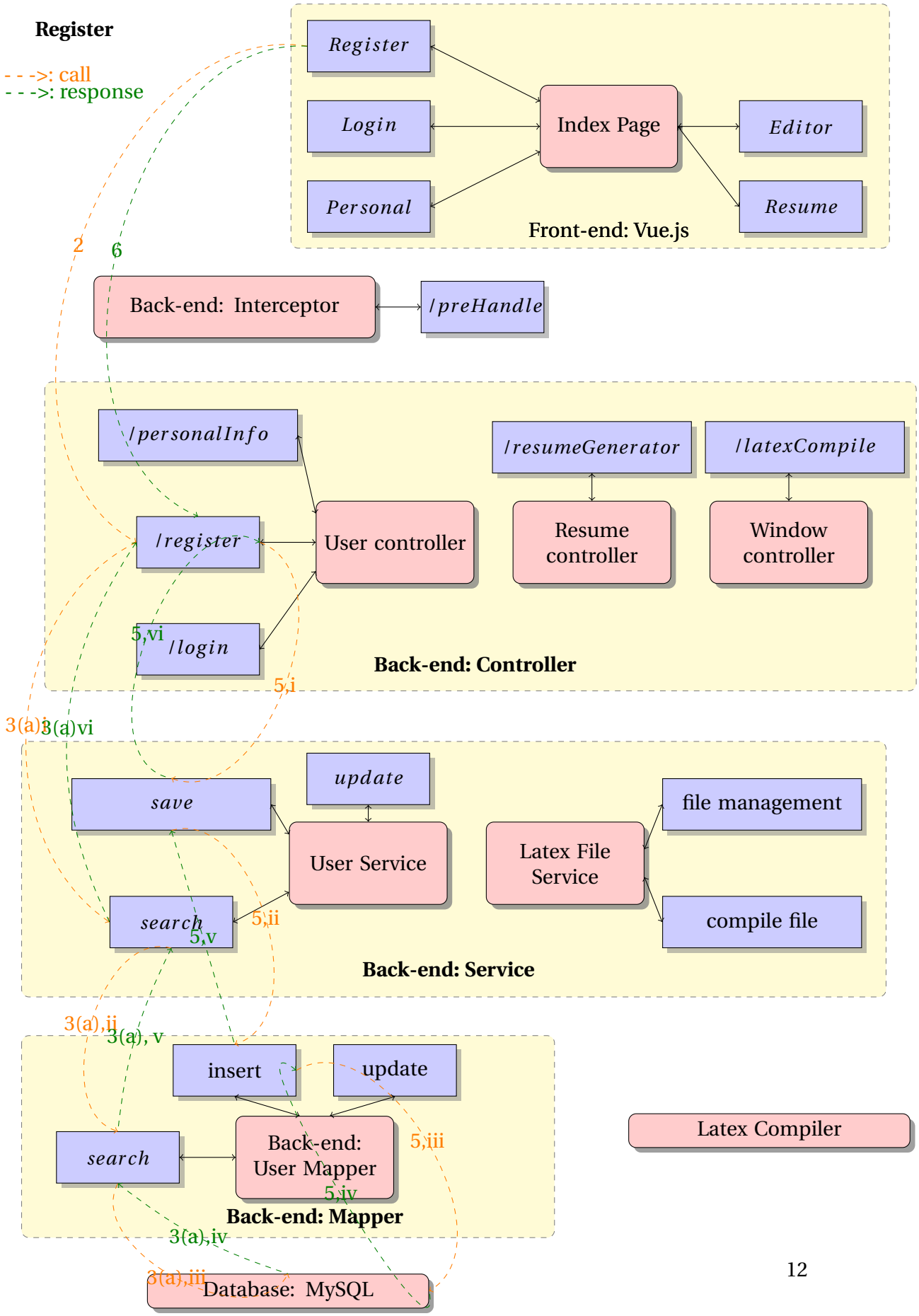
2. When user click "Compile", resume page will apply API to back-end controller "/l atexCompile", note that the data passed to the API is LaTex code
3. "/latexCompile" will pass the information to back-end service: "Latex File Service:file management", , which will:
   (a) call the search service to valid whether the user login or not (if login, the file will be managed to certain file belongs to the user
   (b) generate the latex code and check whether there is code injection
   (c) system call to save the generated latex code and manage the file
4. Then "/latexCompile" will call back-end service: "Latex File Service:compile file", the "compile file" will
   (a) call Latex compiler to compile the latex file
   (b) return the address to the generated file
5. "Back-end:Service:Compile file" will generate an address to the generated file and return the address and other information to "Back-end:windowController:latexCompile"
6. "Back-end:windowController:latexCompile" will response to "Front-end:editor" with the address and some other information
7. "Front-end:editor" will update the url to the generated file and update PDF preview


LaTeX Editor
- - ->: call
- - ->: response
Register
Login
Personal
Index Page
Editor
Resume
Front-end: Vue.js
2
6
Back-end: Interceptor
/preHandle
/personalInfo
/register
/login
User controller
/resumeGenerator
/latexCompile
Resume controller
Window controller
Back-end: Controller
3
3
4
5
update
save
search
User Service
Latex File Service
file management
compile file
3(a),vi
Back-end: Service
3(a),i
3(a),ii
3(a), v
4(a)
4(b)
insert
update
search
Back-end: User Mapper
Back-end: Mapper
Latex Compiler
3(a),iv
3(a),iii
Database: MySQL

Register

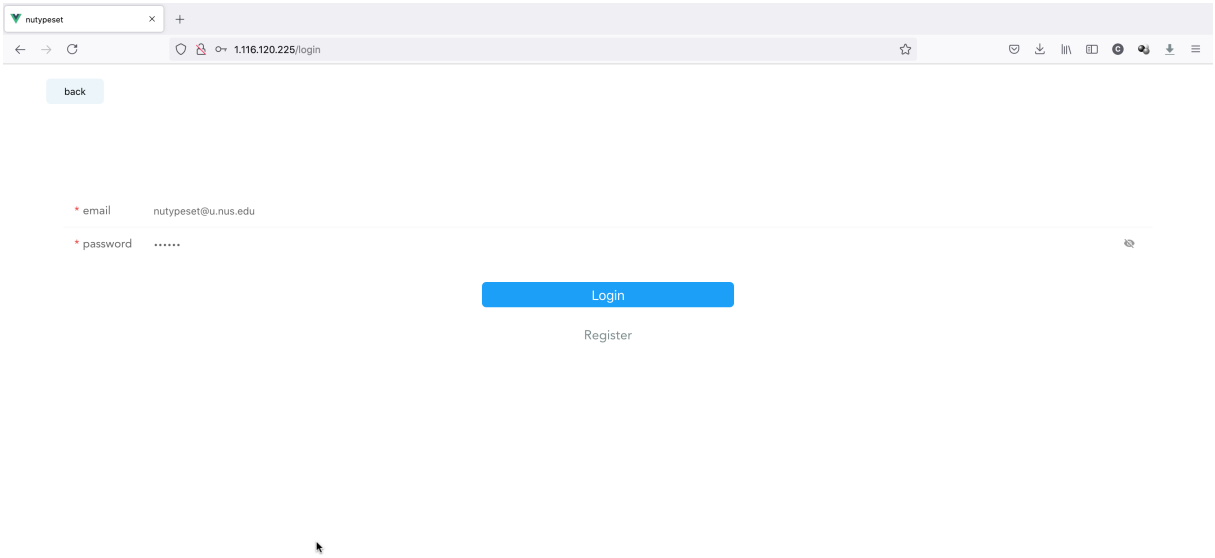
1. In Register page, user should fullfill all compulsory information before submitting
2. When user click "Register", register page will apply API to back-end controller "/user/register"
3. "/user/register" will pass the information to back-end service: "User Service:Search", which will:
   (a) call the search service to see whether the user's email has been registered or not, if the email has been registered return false
4. "back-end: controller: /user/register" encrypts the password with SHA-256 for storage
5. Then "Back-end: controller: /user/register" will call back-end service: "User Service:save" to save the user information in Mysql
6. "Back-end: controller: /user/register" will response to "Front-end:register" with the a boolean value to indicate whether registering successfully or not
7. "Front-end:register" will alert user the register result

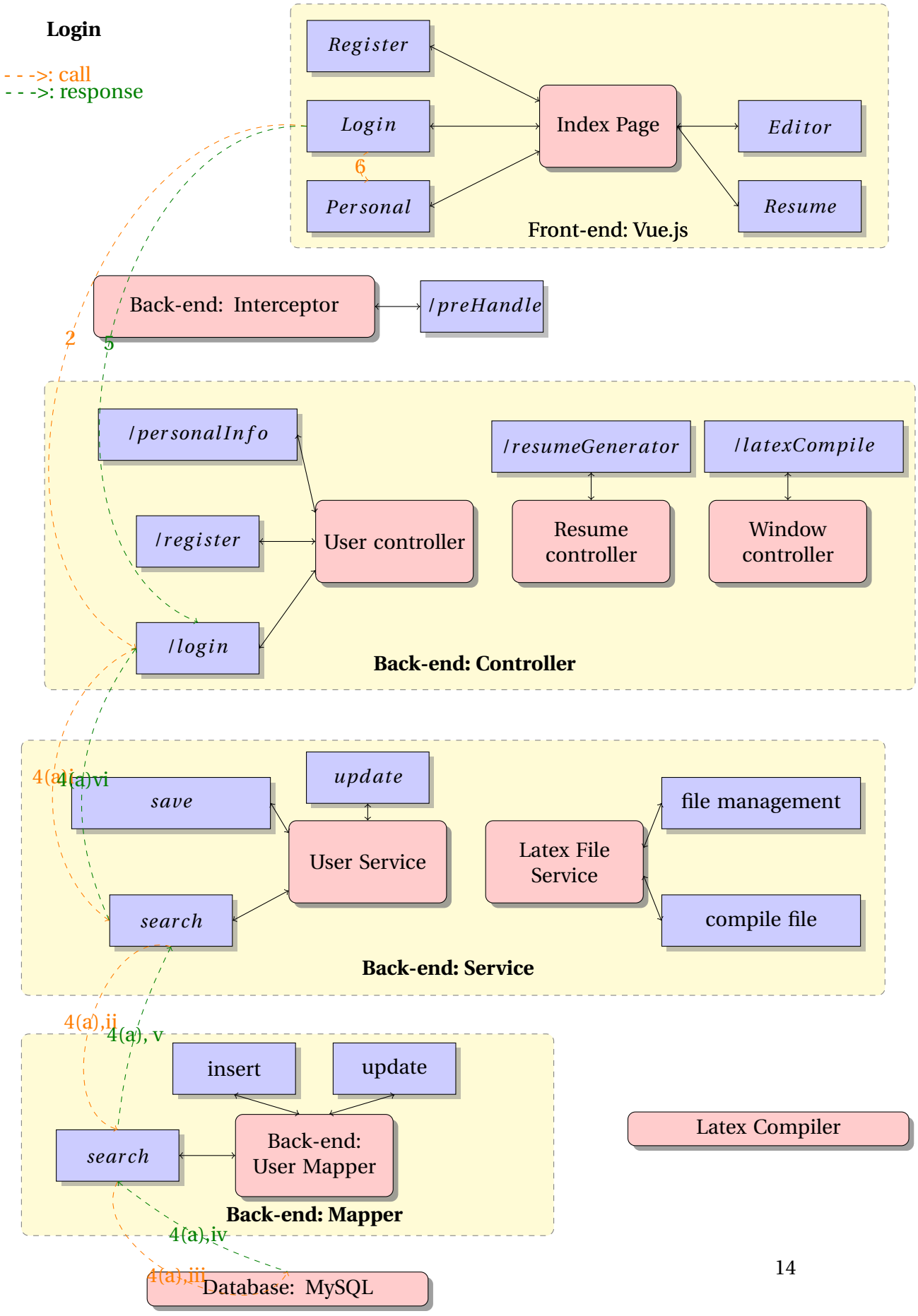



Register
- - ->: call
- - ->: response
Register
Login
Personal
Index Page
Editor
Resume
Front-end: Vue.js
2
6
Back-end: Interceptor
/preHandle
/personalInfo
/register
/login
User controller
/resumeGenerator
/latexCompile
Resume controller
Window controller
5,vi
5,i
Back-end: Controller
3(a)i
3(a)vi
save
update
User Service
search
5,ii
5,v
Latex File Service
file management
compile file
Back-end: Service
3(a),ii
3(a), v
insert
update
search
Back-end: User Mapper
5,iii
5,iv
Back-end: Mapper
3(a),iv
3(a),iii
Database: MySQL
Latex Compiler

Login

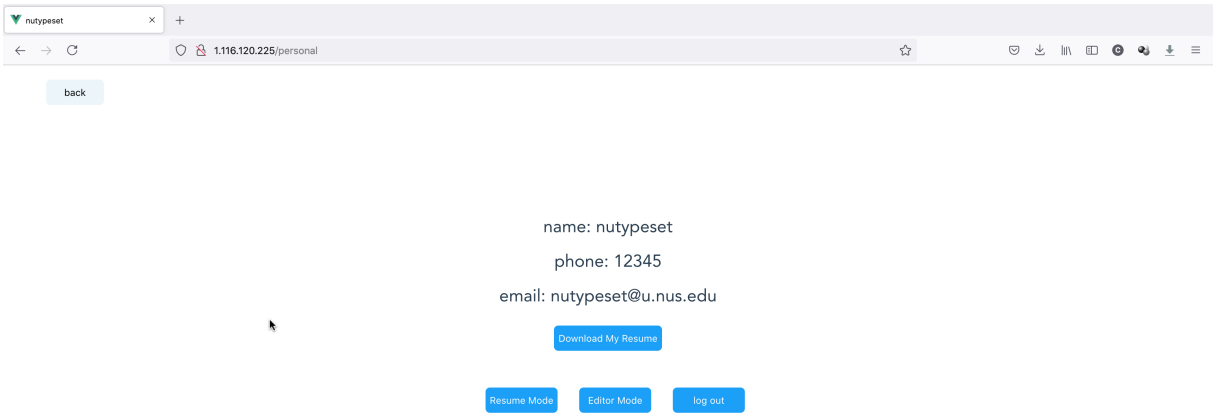
1. In login page, user should fullfill all compulsory information before submitting
2. When user click "Login", login page will apply API to back-end controller "/user/login"
3. "/user/login" will encrypt the password with SHA-256 for query
4. "/user/login" will pass the information to back-end service: "User Service:Search", which will:
   (a) call the search service to see whether there is a user with certain email and hashed password
5. - if there is legimate user, generate a token with JWTUtils and pass true and the token to "back-end:userController/login"
   - if email or password is wrong, pass false to "back-end:userController/login"
6. "back-end:userController/login" will pass the boolean status and the token(if any) to "front-end:Login"
7. "front-end:Login" will
   - if login success, redirect to "front-end:Personal"
   - if login fail, alert user

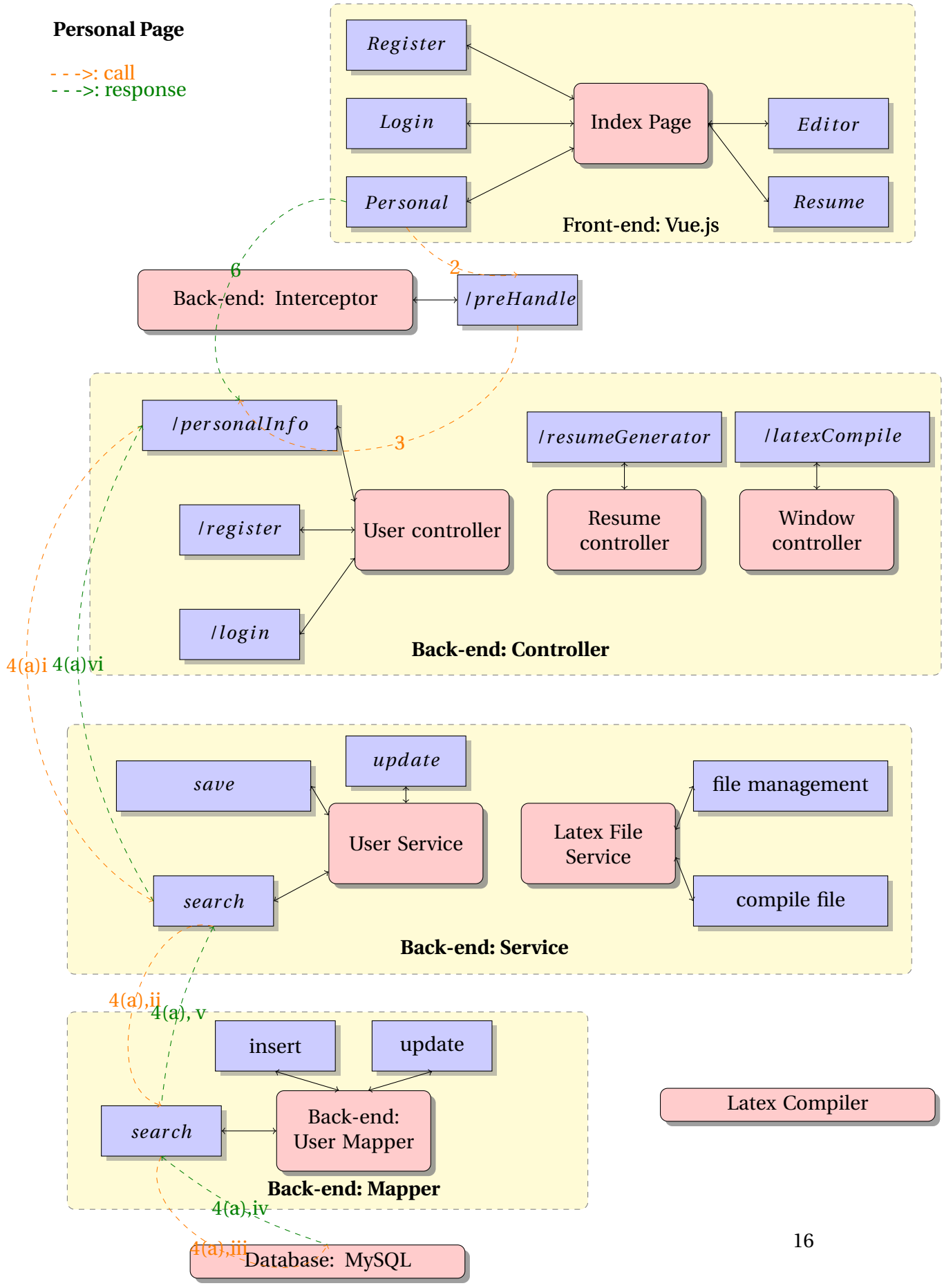



Login
- - ->: call
- - ->: response
Register
Login
Personal
Index Page
Editor
Resume
Front-end: Vue.js
6
Back-end: Interceptor
/preHandle
2
5
/personalInfo
/register
/login
User controller
/resumeGenerator
/latexCompile
Resume controller
Window controller
Back-end: Controller
4(a),i
4(a)vi
save
update
search
User Service
Latex File Service
file management
compile file
Back-end: Service
4(a),ii
4(a), v
insert
update
search
Back-end: User Mapper
Back-end: Mapper
4(a),iv
4(a),iii
Database: MySQL
Latex Compiler

Personal Page

1. Once the user login successfully, the front-end will have the token and user can visit personal page
2. "front-end:personal" will try access back-end api "/personalInfo", but because "/personalInfo" is a private API, token should be firstly passed to "Interceptor:prehandler"
3. Prehandler will decrypt the token, and check whether it is valid, if it is, interceptor will allow the execution of "back-end: userController: /personalInfo"
4. "/user/personalInfo" will pass the information decrypted from the token to back-end service: "User Service:Search", which will:
   (a) call the search service to find the needed information
5. after getting the user's information, "back-end:userController/persionalInfo" will hide some private information such as password and protects the personal information
6. "back-end:userController/persionalInfo" will pass the boolean status and the information
7. "back-end:userController/persionalInfo" will show the personal information if getting the information correctly




Personal Page
- - ->: call
- - ->: response
Register
Login
Personal
Index Page
Editor
Resume
Front-end: Vue.js
6
2
Back-end: Interceptor
/preHandle
/personalInfo
3
/resumeGenerator
/latexCompile
/register
User controller
Resume controller
Window controller
/login
Back-end: Controller
4(a)i
4(a)vi
update
save
User Service
file management
Latex File Service
search
compile file
Back-end: Service
4(a),ii
4(a), v
insert
update
search
Back-end: User Mapper
Back-end: Mapper
4(a),iv
4(a),iii
Database: MySQL
Latex Compiler

Save/update Information

1. update of information is handled by "Back-end:service: update", which may by called by personalInfo, resumeGenerator, latexCompile or some other controller in the futher
2. "Back-end:service: update" should only accept information passed from "interceptor: pre-handler" because identity and authentication are needed
3. "Back-end:service: update" will call "Back-end:Mapper: update" to execute MySQL order

Front-End: Preview

Preview function is available in both editor mode and resume mode. In resume mode user can also preview source code by clicking switch button. HighlightJS is applied for code highlighting. Preview window is fixed on the right hand side, The code and pdf in preview window will be updated by clicking compile button.

Testing

Testing Design

Testing includes self evalutaion part and open testing part. In self-evaluation period, we've tested all the features that we aimed to introduce and fixed several problems, including code highlighting problem in ediotr (Code will not be passed to beackend for compiling after applying highlight feature) and pdf update feature (pdf could not be updated after updating code in editor).

evaluation. Users are asked to explore our web app, then user can generate a resume using resume mode and make further change using editor mode. Users need to finish a survey on Survey Monkey (link is provided on index page).

Testing Result And Analysis

There were about 30 users join this testing. In this testing user are mainly unversity students, and half of them do not have programming experience. The result of testing covers issue from both frontend and backend. Accodring to user feedback, most of users comment that the GUI of NUtypeSet is friendly and easy to use. The majority of users think that it is obligatory to provide upload image, since it is important to have image on resume. Some users comment that with NUtypeSet it is convenient to generate a professional resume and will consider to use it when need in the future. There is also amount of feedback complain that the model of resume on NUtypeSet is too simple. A small part of users complain that the loading is not smooth enough, which affect their experience. Users advise that we could provide more customize options for GUI and optimize our UI as well. For example, preview funciton could be optimal, and personal page is too simple. Overall, NUtypeSet could be better by fixing bugs and optimizing features.

Further Improvement

- More models of resume and varies languages supportc could be provided to satisfy users with different backgrounds.
- Preview window could be optional and resizable.

- Image function will be added later so that user can put their image on resume. (server optimal)
- Linkedin account information syncing could be add as well, so that personal page could be improved as profile, and user can generate different resumes in shorter time.
- AI can also be introduced in order to improve the quality of resume by marking.