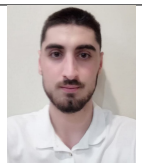
3D Model Binary Vision System

Requirement Specifications Document

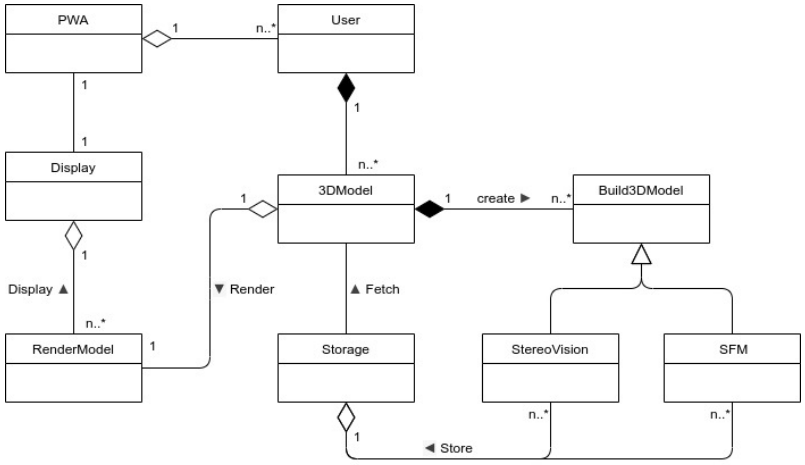
Flap_Jacks

Full Name	Photo
Rani Arraf	
Quinn du Piesanie	
Jacobus Janse van Rensburg	
Steven Visser	
Marcus Werren	

1 Introduction

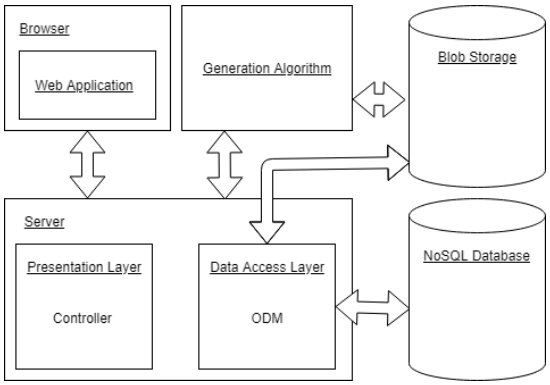
We are designing and making a mobile web application. The main purpose of this system is to scan a patients teeth using a camera, and processing this media into a 3D digital model that can be inspected. The model will then be stored and made available for retrieval at a later stage. The system will also be able to keep track of a doctors patients, as well as each visit that a patient makes to the dentist.

1.1 Domain model



2 Architectural Design

Our System uses a 2-Tier Architecture with a Request-Response Model. Since the system is a web-based application, responses will not be sent to the user unless they have sent a request through the User Interface of the website. The Controller Layer handles all of the users actions, and in turn communicates with the data access and process layers. The Data Access layer communicates directly with the database and accounts for all interactions regarding data retrieval from the database. The Client module is the web application that sends requests to the server module.



3 User Characteristics

As required by the client, there is one main user in mind when creating this project, however, the users may still expand from this.

3.1 Medical Professionals

These professionals are the main target for this project, focusing mainly on professionals in dental health. These users will make use of all of the subsystems provided by the web application, while focusing on the 3D renders of their patients teeth.

4 Functional Requirements

4.1 Use Cases

U1 Login

U2 Sign-up

U3 Add Patient

U4 View Patient

U5 Edit Patient

U6 Edit Details

U7 Logout

U8 Upload Media

U9 Render Model

U10 Inspect Model

U11 Export Model

U12 Record Media

U13 Upload STL

U14 Forgot Password

U15 New Visit for Patient

4.2 Requirements

R1 System must allow users to take their own video input

- **R1.1** The user should be allowed to operate a camera

R2 System must allow users to give prerecorded media as an input

- **R2.1** The user should be allowed to choose what type of media they are providing.
- **R2.2** The user should be allowed to change the media provided.
- **R2.3** The system should upload the selected media to the database.

R3 System must allow users to generate the model

- **R3.1** The system should distinguish the type of media provided.
- **R3.2** The system needs to do all the calculations for the 3D render.
- **R3.3** The system needs to create a mesh from the calculations made.
- **R3.4** The system should send the information of the render back to the web page to be rendered if requested.
- **R3.5** The web page should render the returned data

R4 System should allow the user to inspect a rendered model

- **R4.1** The user should be able to zoom in and out of the render for better perspective.
- **R4.2** The user should be able to rotate the render.
- **R4.3** The user should be able to look around in the rendered view and move to different positions (FPS Mode)

R5 System should be able to store things in the database

- **R5.1** The user should be able to save the render to a database.
- **R5.2** The user should be able to save the media to a database.

R6 System must be able to retrieve information from the database

- **R6.1** The user must be able to retrieve a render data from the database.

R6.2 The user should be able to see the media stored with the render data.

R7 System must be able to export the rendered model

R7.1 Generate a 3D Printable structure

R7.2 Save the structure to the user's device

R8 System should allow users to create an account

R8.1 Allow Users to choose a username

R8.2 Allow Users to choose a password

R9 Allow users to login

R9.1 System should allow a user to log in with valid credentials

R9.2 System should allow a user to logout

R9.3 System should allow a user to reset forgotten passwords

R10 System should allow users to update patients

R10.1 Allow a user to create a patient

R10.2 The user must be allowed to fetch relevant information about a patient such as stored renders of the patients mouth/teeth.

R10.3 The user must be allowed edit and update a patients information for example adding new models of their mouth/teeth.

R10.4 The user must be allowed to add new visits for a patient

R11 System should allow users to search for patients

R11.1 Allow a user to search for a patients name

R11.2 Allow a user to view one of the resulting patients information

4.3 Subsystems

4.3.1 Media System

Subsystem is comprised of the camera that will be used to record media, recording of the media and uploading this or prerecorded media to the database. This media will be input to the render algorithm.

4.3.2 Web Page

This is the platform that the user will interact with the overall system. It encompasses all UI as well as display functionality.

4.3.3 Storage

This subsystem controls all storage and retrieval of information from databases stored on the server.

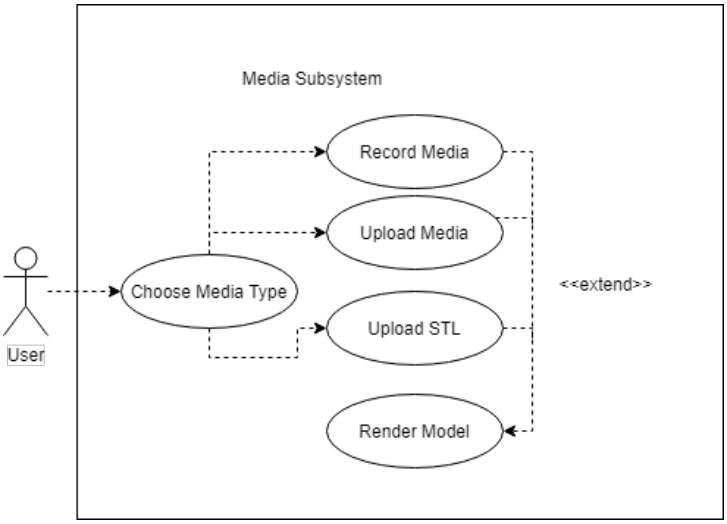
4.3.4 User Management

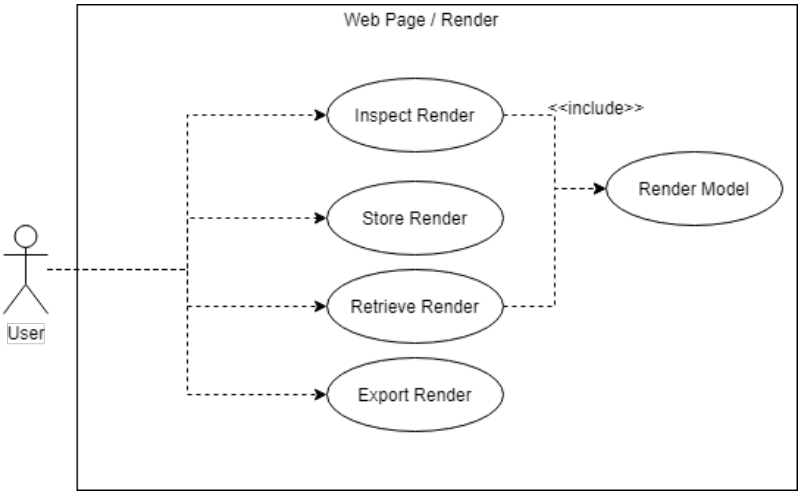
This subsystem encompasses all that is involved with creating users, logging into the system.

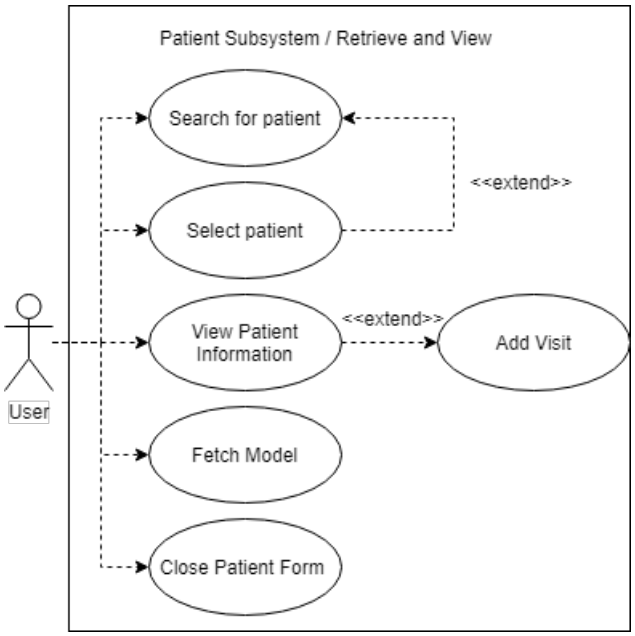
4.3.5 Patient Management

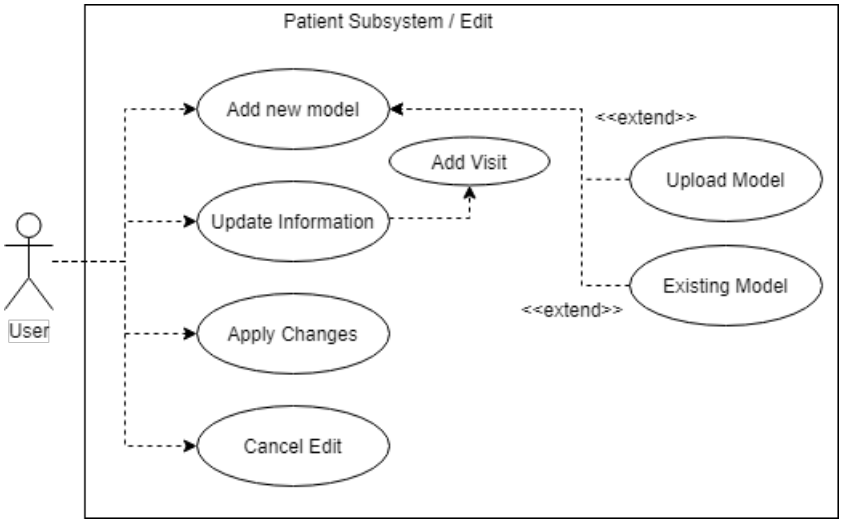
This subsystem encompasses all that is involved with creating patients and editing patient information.

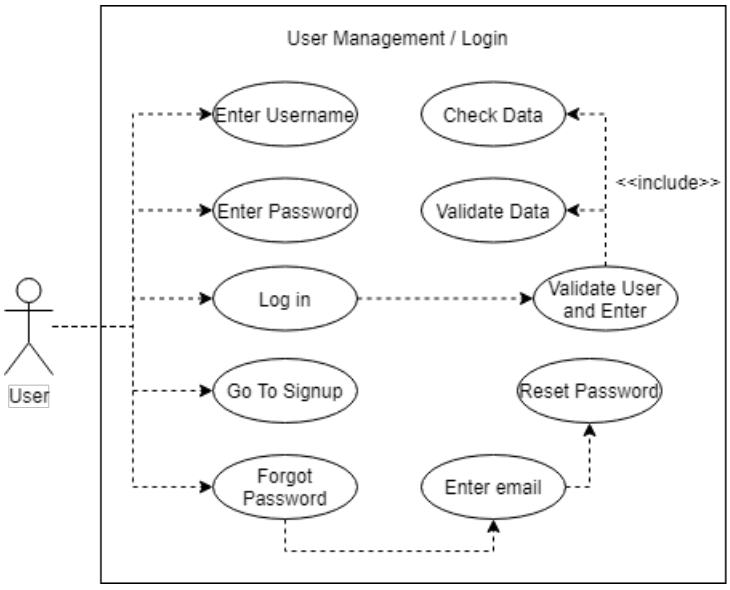
4.4 Use Case Diagrams

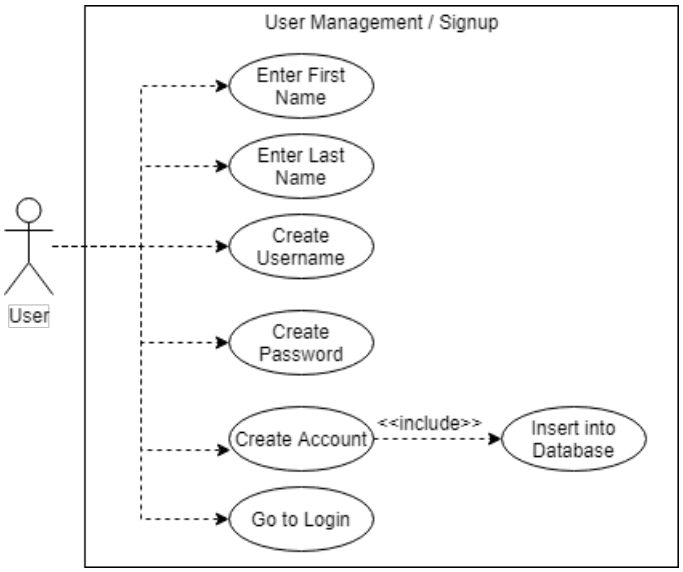





Patient Subsystem / Retrieve and View
Search for patient
<<extend>>
Select patient
View Patient Information
<<extend>>
Add Visit
User
Fetch Model
Close Patient Form


Patient Subsystem / Edit
Add new model
<<extend>>
Add Visit
Upload Model
Update Information
Existing Model
<<extend>>
User
Apply Changes
Cancel Edit





5 Quality Requirements

1 Performance:

The performance is very important in any aspect of the system we are developing. In the case of this project our performance will be the accuracy and the speed at which the render is computed and produced. We ideally would like to

minimize the time taken to create the render and maximize the accuracy of the render. An important aspect to keep in mind is the camera used to take the media photo and the media type. Since the resolution of a camera can greatly impact the speed and accuracy of making the render.

The technology we will be using for this will be three.js for the rendering of the teeth. We will be using JavaScript to import the video data through an API so that the algorithm can easily take in the information and proceed with the rendering. The algorithm itself is coded with C++ because of the libraries it has to offer for the user. Those libraries allow for webgl data integration with the code.

2 Reliability:

One of the biggest frustrations that a user can experience is if an application freezes, crashes or gives incorrect results after spending precious time waiting for it to do its computations. We aim to prevent the application from breaking as much as we possibly can and to ensure that our results are accurate 95% of the time. To ensure that the application will not break, the server in our Client-Server architecture will handle all of the processing of the users input when it comes to rendering a model. This is so that the user device does not have to have the high computational power that is required to ensure results are generated and returned as quickly as possible without the application crashing from over-usage of RAM. We used MongoDB as this database integrates well with a webserver, which allows for data to be transferred and returned when requested from the server to the application. We also used restful API calls so that the calls to retrieve information are made to the server and the server then returns the required information to the client.

3 Security:

To demonstrate how the doctors will use the system the user will be required to log in. The user will therefore require a username and a password in order to use the system. This login information will be used when storing the media and the render info. This will ensure that we know which Doctor stored which information and prevent other doctors to be able to retrieve information that does not belong to them which will be a violation of doctor-patient-confidentiality.

The technology we used for this system is JavaScript and hash coding systems that converts the password in a non readable state. This will prevent malicious users from ever brute forcing the system. We will be using SSL encryption in the future to prevent any form of 'man-in-the-middle' attacks. Therefore, this will ensure that the data will end up being 93% secure. (40% from hashing; 30% correct data creation; and 23% SSL encryption)

4 Maintainability:

Maintainability refers to the ability of the system to be easily repaired. The system will be divided into smaller subsystems that will each be able to only do their individual tasks. Therefore if something breaks it will be easy to find the error in its subsystem. We will employ appropriate error messages so that the developer knows where and why the error occurred. With proper unit/integration tests it will be easy to monitor the working state of the system.

With the use of continuous integration software (Travis CI) successful repair action will be made easy. The tools used will ensure that meaningful commits are made to the repository by all active developers working on the project in parallel. The tools used will show developers whether their commits contain errors and are successful commits. If the commits contain errors, these errors can be quickly fixed and then rechecked relatively quickly by the developer ensuring that the code its maintained properly and working correctly.

Above Travis CI, we used a coding standard that places all relevant code in subsections. This allows for easy editing and manipulation of the code. With this in mind, it helps produce a maintainable system that offers system surveillance and system integrity.

5 Usability:

The usability of a system is measured on how easy it is to use the product. Our system is usable after focus and effort has been placed in the fields of learnability, efficiency , satisfaction and error control.

Our system is easy to learn due to the linear and simple layout of the control flow and user actions. The learn-ability of the system is also enhanced with the minimal layout that shows clear action labels and proper headings. The system is also really memorable due to its unchanged design and easy flow of the layout. The system is efficient since the user can visually see the changes that happen to reach their goals. This is due to the atomic nature than we implement each user action to be able to see most changes as soon as they occur. We minimise the errors that a user can create by implementing verification's in both the front and back end to prevent system errors. The UI has been designed to feel a familiarity related to medical experts. Our system uses forms that can easily be understood.

The technology we used to create the usability of the system is HTML, Javascript and Node js. We implemented the architecture as a Client-Server architecture to handle all the user-requests. We use a Model-View-Controller in our API to handle the requests that the user makes and returns the results that the front end javascript receives and process for visual output to the users webpage.

6 Trace-ability Matrix

	SS1	SS2	SS3	SS4	SS5
R1	x				
R2	x				
R3	x	x	x		
R4		x	x		
R5	x		x		
R6	x	x	x		
R7			x		
R8			x	x	
R9			x	x	
R10			x		x
R11		x	x		x
QR1	x	x			
QR2	x	x	x	x	x
QR3			x	x	x
QR4	x	x	x	x	x
QR5		x		x	x