Camagru MVC Design:

Creator: Henricus Wouter de Vos (Slack: @hde-vos)
Supervisor: Abduraghmaan Gabriels (Slack: ~@agabrie)
If you want to join in on edit/reviewing/scrutinizing the document, please slack @hde-vos *:D*

Note:

*Remember to read everything. All data in this document has been written so that it is relevant and accurate to the PDF's requirements.
*Guest: not logged in user
*All abbreviations will have a single instance where their full wording will be shown. After that, it's up to you to remember it.
*All areas were data is being received from the user/guest has to be protected against SQL Injection and uploading unwanted content. (Don't know what this is? Read Camagru's Specifications)
*All accounts that have not been verified (Meaning that they have verified their account through a link sent to their email) will not be able to log in, post images or write comments.

For the design, we're taking a very school-ly type approach in breaking down Camagru. We have 3 parts to the project: Model, Visual and Controller. Visual will be the easiest for us. All of it is laid out in the PDF and my other Documentation, Camagru's Specifications.

I'll be working on my Visuals first. We start here, because this is the known. Once we have all the visuals down, we can start discerning what functions we'll need for the Visuals to communicate with the database.

Our next step will be designing the database. I'll be following a relational database format.
See: Relational Databases, Primary keys, Foreign Keys, More Primary and Foreign Keys
I'll have 1 database with 4 tables. A user table containing user information, a gallery table containing encoded images, a comment table holding comments for each image and a likes tables that can be used to determine the count of likes and the person who liked a photo.

Information from these tables will be formatted as follows: database.table.column, however, as we are only using a single database, I'll be dropping the "database." part. So typical references to the database's data will look like this: user.username, user.password, user.id, image.id, image.source, image.owner, comments.owner, comments.id, comments.text.

Lastly will come our Controller. Our Controller will take info from the Visual and store the data in the Model. It will also take data from our Model and display it to the Visual. We have no idea what we'll need to do here until our Visual and Model are done or close to done.

ANOTHER NOTE: Validation and Verification have different meanings. Validation means “Can this be a correct input?” and Verification means “Is this the right input?”. You will validate a password to see if it contains uppercase, lowercase and special characters and that it’s longer than 8 characters. You’ll verify an existing user’s password by checking if the given password matches the password in the database. Please make sure you understand these two definitions well, as they’ll be used often.

Visuals:

Starting with the visuals ie. HTML, CSS and Javascript. To make the pages look good, make a header, footer and body for every page. This allows us to copy-paste most of the page layout and we won't need to make a new page for every section. The only page that'll be different will be the Editor page. It requires multiple "body" panels, so some extra effort will be needed to meet the requirements. HTML Events like "onclick" and "resize" will not be explained

More notes: *This may not be the final draft, I may have forgotten some pages.
*User needs to be able to log out on any page (This is a requirement)
*Every page will have a Drop-down Menu containing links to their account details, Gallery and Editor. (This is not a requirement, but it is my preference and most of this I'll copy-paste, so not much extra effort here.)
*All bonus content is preceded with a " ^ ".
*^All pages will have a profile picture in the top right corner, inside the header.
*The footer has to contain our "@*username*" trademark somewhere (~@agabrie told me this, can't find anything related to it in the PDF).

We need the following pages, each with the following content:

1. **Signup**
2. **Sign In**
3. **Account Confirmation**
4. **Password Reset**
5. **Account Details**
6. **Gallery**
7. **Editor**
8. **Verify Email**
9. **Profile**
10. **Users**
11. **^Chat**

Signup: This page is where a user creates an account to use our Camagru website.

- Textbox for Username
- Textbox for Password
- Textbox for Email
  (We'll need validation on all these textboxes. I'll split them into ValidateUsername, ValidatePassword and ValidateEmail. These are the first 3 functions we know we'll need.)
- Button to Create account - "onclick" the user should be prompted on whether the account creation succeeded or failed. If succeeded, it should take them to a page that tells them to verify their account through their email. User's still can log in.
- Drop-down Menu

Sign In: This page is where a user signs in to an existing account.

- Textbox for Username (UserVerification will be required)
- Textbox for Password (PasswordVerification will be required)
- Button to Sign in (Should be disabled if any textbox is empty and redirects to Gallery.html)
- Button to Create account (Redirects to CreateAccount.html)
- Button to Reset password(Redirects to ResetPassword.html)
- Drop-down Menu

Account Confirmation: This page will only be seen when a user receives an email linking them to it. Going to this url will mean that the user's account will be verified.

- Text that will tell them to click on the button to be verified.
- Button Confirm Account to confirm account creation and logs user in (Redirects the user to the Profile.html)
- Drop-down Menu

Email Password Reset: This page will be seen when the user clicks on the "Send Password Reset Email" button on the signup page.

- Textbox for the email (No validation here, we'll simply display the same message whether a user exists or not, else existing usernames can be found and brute forced. See: Brute Force Attack)
- Button to Send Password Reset Email
- Button to go back to sign in page
- Drop-down Menu

Reset Password: This page will be seen when the user clicks on the reset password link sent to their email.

- Text that will tell them to click on the button to reset password.
- Textbox for the new password (PasswordValidation will be required)
- Button to Reset Password

Account Details: This page will be used to change profile information such as the username, password, email and notification settings.

- Button to Immediately log out
- Textbox for new Username (UsernameValidation will be required)
- Textbox for new Password (PasswordValidation will be required)
- Textbox for new Email (EmailValidation will be required)
- Textbox for current Password (PasswordVerification will be required)
- Option to disable Notifications via Email
- Button to update account details
- Drop-down Menu
- ^File section for user to upload profile picture

Gallery: This page will show the 5 newest images from the "Images" Table.

- 5 Images on the page
- Button to go to next 5 images
- Button to go to previous 5 images
- Drop-down Menu

Editor: This page will allow the user to superimpose 2 images over each other. Images should be retrievable from the Webcam or uploaded from the user's device.

- Guest user's should be gently rejected from using the editor
- Button to immediately log out
- Button to upload image
- Button to take picture through webcam
- Button to save current image in the webcam panel.
- Panel that shows a preview of the webcam
- Panel that show a list of superposable images
- Panel that shows all of the user's past posted images
- Drop-down Menu

Verify Email:

- Show text that tells a user to Verify their account by clicking on the link that was sent to their email.
- Button that redirects them to Gallery

Following pages are quite complex, so I don't have a clear idea of how I'm going to implement them. Will update this as I do research and figure it out.

Users:

- Is a page that will display a specific users images, details and a ^button to contact them.

Profile:

- Is a page that will display the logged in users images, details and a ^Drop-down that shows all previously messaged users.

^Chat:

- Textbox for showing past sent messages.
- Textbox for writing a new message.
- Button to send message.

Model and Controller:

Now that we have the rough GUI established, we can determine what functions we'll need for the GUI to interact with the database (Model). This is the PHP section. We'll use each page in the previous section as a guide for the functions. All functions in *Italics* will be functions that we've already written or is close to another functions and can be reused. Functions will have "activators" and "parameters". Buttons/onclick events are "activators" and textboxes are "parameters". Functions will be formatted like the following: *function(textbox."lorem", textbox."ipsum", textbox."dolor", button."sit")*. This means that button.sit will call/"activate" the function and *textbox."lorem", textbox."ipsum" and textbox."dolor" are the parameters that are passed to the function.*

We need functions for the following pages:

1. **Signup**
2. **Sign In**
3. **Account Confirmation**
4. **Password Reset**
5. **Account Details**
6. **Gallery**
7. **Editor**
8. **Verify Email**
9. **Profile**
10. **Users**
11. **^Chat**

Signup:

- ValidateUsername(textbox.username) - Will take in the username and check whether it conforms to your own validation requirements. Returns TRUE if validation passes, else returns False.
- ValidatePassword(textbox.password) - Will take in the password and check whether it conforms to your own validation requirements. Returns TRUE if validation passes, else returns False.
- ValidateEmail(textbox.email) - Will take in the email and check whether it contains a "@" and a ".". Returns TRUE if validation passes, else returns FALSE.
- StoreUserDetails(textbox.username, textbox.password, textbox.email, button.createaccount) - Will store the details retrieved from the textboxes and store them in the "users" table, if they all pass their own validations. VerifyUser(textbox.email) will be called.

  Other things that will be given default values: "Verified" is a column in the table "user" with a boolean that will be set to false.

- VerifyUser(textbox.email) - Will send an email to a user with a link to verify their account.

Sign In:

- VerifyUsername(textbox.username) - Will look in the “users” for the inserted username. If it finds a corresponding username, if will retrieve the “user.id” from the “users” table and call VerifyPassword().
- VerifyPassword(textbox.password) - Will look if the inserted password matches the password of the user at “user.id”.
- Login(button.signin) - If both of the “Verify” functions pass, then the user will be logged in and will store all the users details in the $_SESSION variable.

Account Confirmation:

- ConfirmAccountCreation(button.confirmaccount) - This function will set the “Verified” boolean value in the database to True.

Email Password Reset:

- PasswordResetEmail(textbox.emailpasswordreset, button.sendpasswordresetemail) - Will send an email to a user containing a link where the user can reset their own password.

Reset Password:

- ResetPassword(textbox.newpassword, button.resetpassword) - This function will replace the user’s old password with the new password.

Account Details:

- UpdateAccountDetails(textbox.newusername, textbox.newpassword, textbox.newemail, checkbox.notifications, ^file.profilepicture, button.updateaccountdetails) - This function will update the user’s old username, password, email, notifications settings and ^ set the user’s new profile picture.

Note: Gallery and Editor is territory where I am not 100% of how I’m going to implement them. Therefore, most of it will be more vague than the past view pages. Once I have successfully implemented these features, then I’ll update this section.

Gallery:

- Object “Post” will be used to format the page. See the Objects Category below.

Editor:

- Preview Webcam Footage
- Panel that lists all superimposable images.

- Panel that shows all previously posted images of the logged in user.
- TakePicture(image.webcam, button.takepicture) - Will capture the current image on the Webcam Preview.
- MergeImage(image.webcam, button.savepicture) - Will merge all superimposed images into one image and post it on the user's profile and the Gallery.

Verify Email:

- UserVerified(button.gotogallery) - Will redirect user to the Gallery Page.

Users:

- Object "Post" will be used to format the page. See the Objects Category below.

Profile:

- Object "Post" will be used to format the page. See the Objects Category below.

^Chat:

- SendMessage(textbox.newmessage) - Will store message being sent into the database.
- ShowMessages(textbox.messages) - Will display previously sent messages.

General Functions:

- Logout(button.logout) - This will clear the $_SESSION variable and redirect them to the Galler.html. They will now be a "Guest".

Objects:

- Object "Post" has the following Attributes {
  User.id, user.username,
  user.profilepicture,
  image.creationdate,
  image.source,
  ArrayOf[show comments where comments.foreignkey = image.primarykey],
  image.likes }

The following errors depend on how your phpmyadmin is setup, however this is how I fixed them with my setup of phpmyadmin:

*SQLSTATE[HY000] [1130] Host '127.0.0.1' is not allowed to connect to this MySQL server, then run the following SQL command in phpmyadmin "UPDATE mysql.user SET

Host='%' WHERE Host='localhost' AND User='root'; FLUSH PRIVILEGES;" at this url.
Remember to edit your port number if it is not 8080.

*SQLSTATE[HY000] [2002] No such file or directory, then change your mysql:host=localhost; to mysql:host=127.0.0.1; or Vica Versa.

Model:

The following section will be the Model or Database section. The easiest of the 3. Remember to use a standard for your database, table and column naming convention. I'd suggest keeping everything in lowercase and make sure the column's name is RELEVANT ie. a "user" column is a SINGLE user's information, therefore we call it "user", singular. In our images table, we call a column "likes", PLURAL, because there are multiple. Using random names WILL cause you hours of frustration, try to be as accurate when naming databases, tables and columns.

*tinyints are basically Booleans.
See: Store Images as Blob Datatype, Database's types of ints

"user" table:

Column	user.id	user.username	user.password	user.picturesource	user.verified	user.email
Datatype	int - auto increment	varchar	varchar	varchar	tinyint	varchar
Max Length	None	25	255	255	1	100
Can be NULL	No	No	No	Yes	No	No
Hashed	No	No	Yes	No	No	No
Encoded	No	No	No	No	No	No
Primary Key	Yes	No	No	No	No	No
Foreign Key	No	No	No	No	No	No

"image" table:

Column	image.id	image.source	image.creationdate	image.userid
Datatype	int - auto increment	varchar	DateTime	int - auto increment
Encoded	No	No	No	No
Can be NULL	No	No	No	No
Primary Key	Yes	No	No	No
Foreign Key	No	No	No	Yes

“comment” table:

Column	comment.id	comment.userid	comment.imageid	comment.text
Datatype	int - auto increment	int - auto increment	int - auto increment	varchar
Max Length	None	None	None	100
Can be NULL	No	No	No	No
Primary Key	Yes	No	No	No
Foreign Key	No	Yes	Yes	No

“like” table:

Column	likes.id	likes.userid	likes.imageid
Datatype	int - auto increment	int - auto increment	int - auto increment
Can be NULL	No	No	No
Primary Key	Yes	No	No
Foreign Key	No	Yes	Yes