

C242-PS469

bangkit

Product-Based Capstone



2024

Our Team

Mobile Development - Team



Sukma Wati

Machine Learning - Team



Abd Rahman Wahid Salsabila Putri

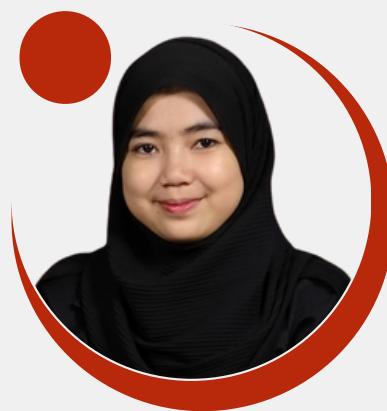


Muhammad Hidayatul
Fadillah

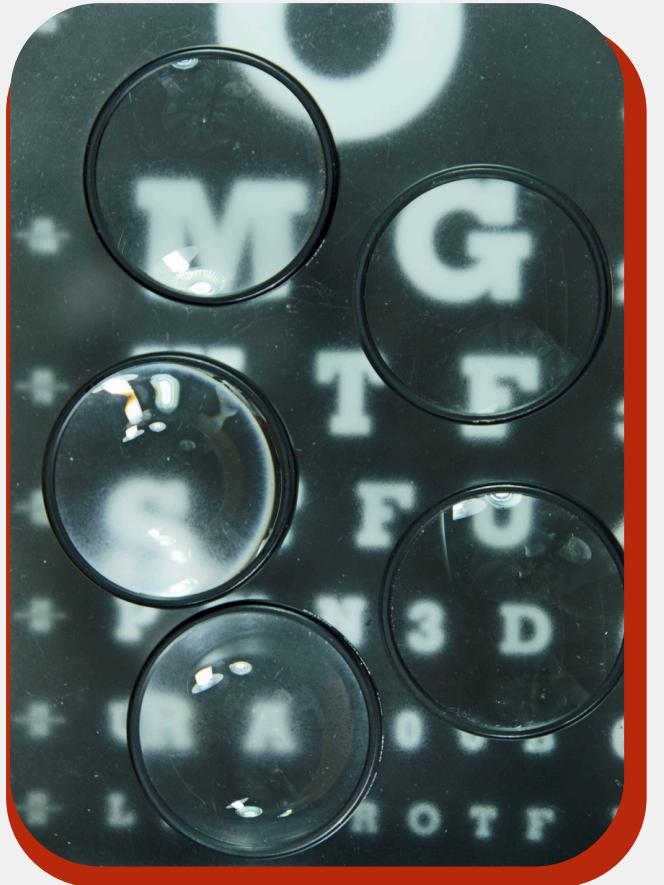
Cloud Computing - Team



Ahmad Faisal



Galbi Nadifah



Background

Cataracts and refractive errors (such as myopia, hyperopia, and astigmatism) are significant contributors to visual impairment worldwide. According to the 2019 **WHO report**, over 2.2 billion people are affected by vision problems, with cataracts being the leading cause of blindness. Despite advances in eye care, many individuals in underserved communities lack access to timely diagnosis and treatment due to the high cost and limited availability of healthcare facilities. Early detection and regular monitoring are crucial in preventing severe vision loss, yet these remain out of reach for many. With increasing **global awareness** of eye health, there is a pressing need for accessible, affordable, and accurate solutions that empower individuals to take control of their vision care.

Reason

The idea of creating OptiLens stems from the urgent need to address the challenges faced by individuals with limited access to ophthalmological resources. Inspired by technological advancements, we believe that machine learning and image processing can revolutionize eye health management. OptiLens intrigued us because it provides a practical solution to improve eye care accessibility while maintaining medical relevance.

By integrating cataract detection through advanced algorithms and manual refractive clarity assessments, our application bridges a critical gap in eye health diagnostics. Our goal is to make early diagnosis and vision monitoring accessible to vulnerable communities, ultimately reducing preventable blindness and empowering individuals to take proactive steps in managing their eye health.



Existing Result

C242-PS469

 bangk!t

Previous research and innovations have tried to utilize technologies such as Machine Learning and cloud-based applications to detect eye diseases, including cataracts. Some examples are:

Mobile Vision Health Apps

Several mobile apps have provided image-based diagnosis for early detection. However, many only focus on a single feature, such as cataract detection without providing additional insights, such as educational articles or manual refraction prediction.

Optimal Machine Learning Model

- Using CNN architecture with data augmentation enriches the variety of datasets, making the model more adaptive to different image conditions.
- The model is optimized using early stopping and hyperparameter tuning to achieve high accuracy without overfitting.

Mobile development

- The UI/UX prototype was designed using Figma to provide a simple and interactive user experience.
- The application development is based on Android Studio, supporting direct image upload from camera or gallery.

Cloud Computing

- Using CNN architecture with data augmentation enriches the variety of datasets, making the model more adaptive to different image conditions.
- The model is optimized using early stopping and hyperparameter tuning to achieve high accuracy without overfitting.

Why Choose a Specific Implementation/Improvement?

- Cost Efficiency
- Ease of Replication
- Transparency

MACHINE LEARNING

Result

C242-PS469



DATASET PREPARATION

- The dataset was categorized into three classes: Immature, Mature, and Normal. The data is divided into training, validation, and testing.
- Data augmentation was performed to enrich the variety of training data using ImageDataGenerator.

MODEL ARCHITECTURE

- Use TensorFlow to build Model
- Convolutional Neural Network (CNN)
- The model uses Sequential API with layers

MODEL TRAINING

- Optimizer: Adam.
- Loss Function: Categorical Crossentropy.
- Metrics: Accuracy.
- Callback: Early stopping is used to stop training when the model does not improve.

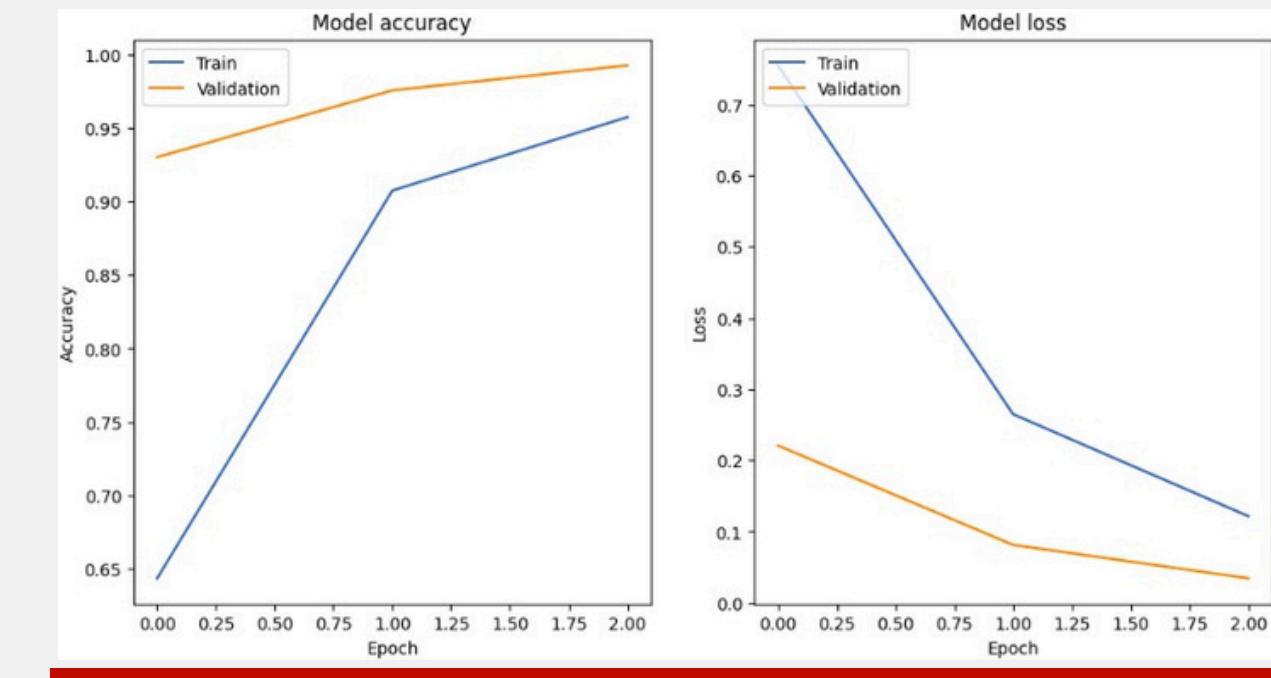
TRAINING RESULT

```
Epoch 1/20
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:122: UserWarning: Your
    self._warn_if_super_not_called()
83/83      118s 1s/step - accuracy: 0.5227 - loss: 0.9152 - val_accuracy: 0.9149 - val_loss: 0.1781
Epoch 2/20
83/83      113s 1s/step - accuracy: 0.8841 - loss: 0.2865 - val_accuracy: 0.9868 - val_loss: 0.0612
Epoch 3/20
83/83      0s 1s/step - accuracy: 0.9434 - loss: 0.1551
Akurasi Validasi telah mencapai: 0.9905, menghentikan pelatihan.
83/83      140s 1s/step - accuracy: 0.9434 - loss: 0.1552 - val_accuracy: 0.9905 - val_loss: 0.0461
```

VISUALIZATION OF RESULTS & MODEL SUMMARY

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_1 (Conv2D)	(None, 61, 61, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_2 (Conv2D)	(None, 28, 28, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 128)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 128)	3,211,392
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 3)	195
<hr/>		
total params:	9,939,275	(37.92 MB)
trainable params:	3,313,091	(12.64 MB)
non-trainable params:	0	(0.00 B)
optimizer params:	6,626,184	(25.28 MB)

Model Architecture: CNN model structure with a total of 9.9 million parameters, consisting of convolution, pooling, and fully connected layers.



Loss graphs and accuracy: The graph shows an increase in accuracy and a decrease in loss for training and validation, indicating the model is learning well.

OptiLens

Use Technology



TensorFlow



Google
Colab



OpenCV



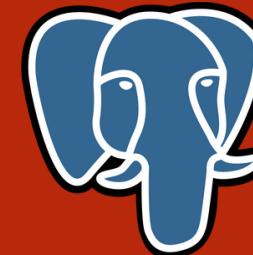
PIL
(Python Imaging
Library)



NestJS



PostgreSQL



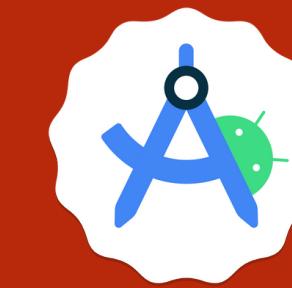
TypeScript



Docker



Figma



Android Studio



Kotlin

- 1** Build a modular and scalable backend architecture with NextJS and TypeScript-based Framework
- 2** Employing PostgreSQL for reliable and high-integrity data management
- 3** Leveraging Google Cloud Platform for deployment and scaling
- 4** Isolated application environments and ensuring consistent with Docker
- 5** And Create some APIs for integration to Mobile Development

CLOUD COMPUTING

Plan

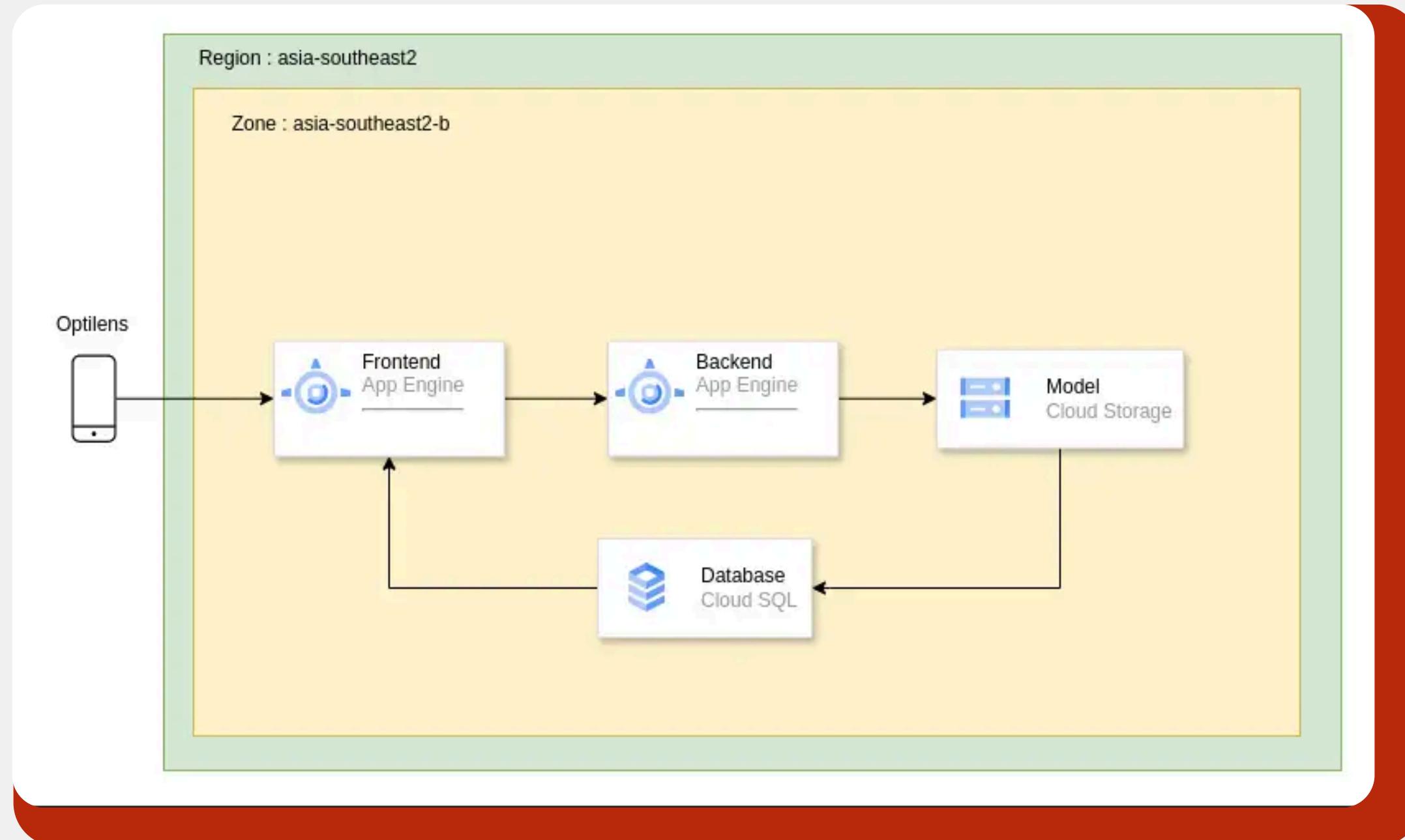
Budget & Expenses (GCP Cloud) :
\$50

CLOUD COMPUTING

C242-PS469

Cloud Architecture

bangkit



Auth

Auth	
POST	/api/auth/login
POST	/api/auth/refresh-token
POST	/api/auth/register
GET	/api/auth
PUT	/api/auth/reset-password

User

user	
GET	/api/user Search users
GET	/api/user/{id} Get user details
DELETE	/api/user/{id} Delete user
PUT	/api/user/edit-profile/{id} Edit user profile for admin
PUT	/api/user/edit-profile Edit user profile

Artikel

Artikel	
GET	/api/artikel Retrieve a list of articles
POST	/api/artikel Create a new article
GET	/api/artikel/{id} Retrieve details of a specific article
PUT	/api/artikel/{id} Update an existing article
DELETE	/api/artikel/{id} Delete an article
GET	/api/artikel/{id}/gambar Retrieve the image of a specific article

Katarak

Katarak	
POST	/api/katarak/predict
GET	/api/katarak/history/{id}/image
GET	/api/katarak/history

API from CLOUD COMPUTING

PROTOTYPING
THE UI/UX

CREATE AND SET-UP
THE PROJECT ON
GITHUB

IMPLEMENTING THE
PROTOTYPE INTO
THE PROJECT

DESIGNING
ARCHITECTURE
COMPONENTS

APPLY INSTRUMENTAL AND
UNIT TESTING, BUG FIXING
FOR FINAL APP

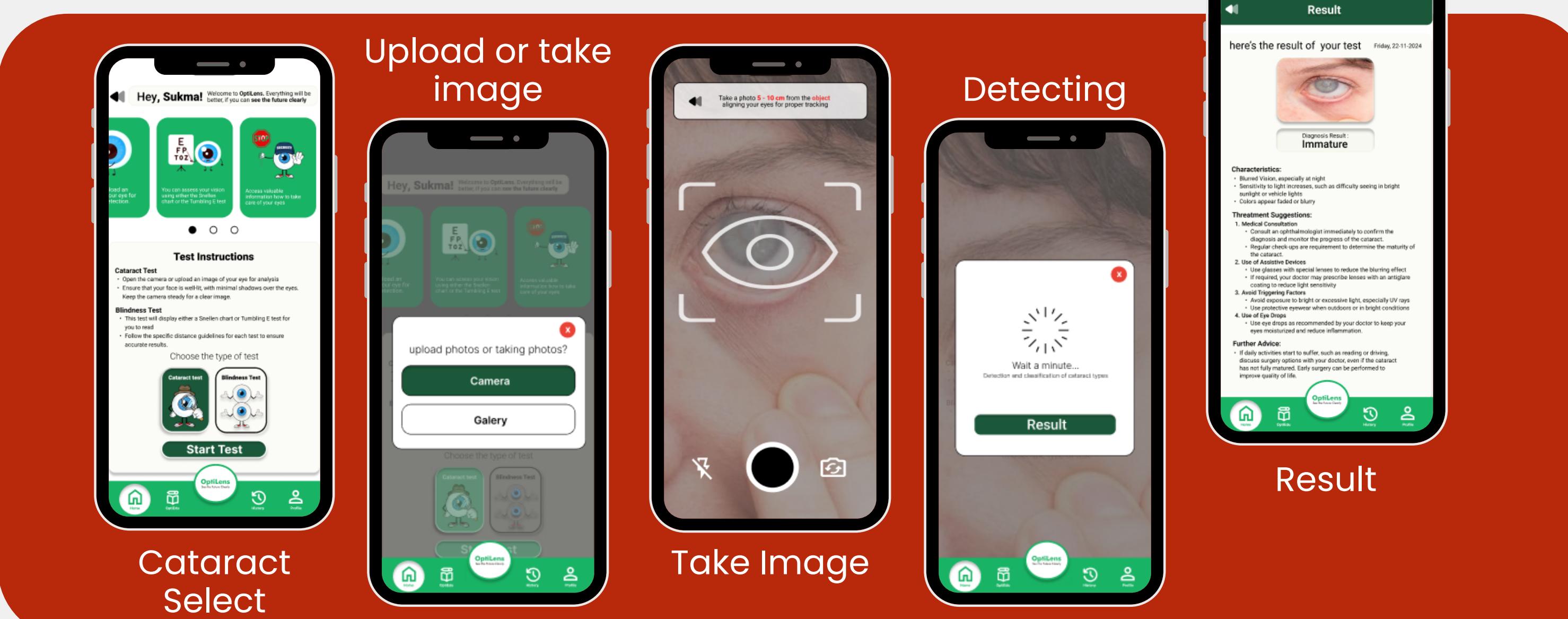
MOBILE DEVELOPMENT

Plan

The image displays five mobile application screens for the OptiLens app, arranged in a grid-like layout:

- Welcome Guide:** Shows the initial screen with a green and white abstract background, featuring the OptiLens logo and the tagline "See the Future Clearly". A large green button labeled "GET STARTED" is at the bottom.
- Login:** Shows the login screen with fields for "Username" and "Password". It includes a "Forgot Password?" link and a "LOGIN" button. Below the buttons is a "Continue with Google" option. At the bottom, it states: "OptiLens is a mobile application for self-diagnosis of cataracts and blindness".
- OAuth:** Shows the OAuth login screen for "Login with Google". It displays three account options: "Account-1", "Account-2", and "Account-3", each with an email address. Below the accounts is a note: "Before using this app, you can review OptiLens' privacy policy and terms of service".
- Sign Up:** Shows the sign-up screen with fields for "Username", "Email", and "Password". It includes a "SIGN UP" button. Below the buttons is a note: "Don't have an account? Login".
- Main Menu:** Shows the main menu screen for "Hey, Sukma! Welcome to OptiLens...". It features three main sections: "Scan or upload an image of your eye for cataract detection", "You can assess your vision using either the Seelies chart or the Tumbling E test", and "Access valuable information how to care of your eyes". Below these sections is a "Test Instructions" section with details for "Cataract Test" and "Blindness Test". It also includes a "Choose the type of test" section with "Cataract test" and "Blindness Test" options, and a "Start Test" button. The bottom of the screen has navigation icons for Home, OptiLens, History, and Profile.

Mobile Apps Design



The image shows a sequence of five mobile application screens for the OptiLens app, illustrating the process of detecting cataracts:

- Cataract Select:** The user selects the "Cataract Test" option. The screen displays instructions: "Load an eye for inspection," "You can assess your vision using either the Snellen chart or the Tumbling E test," and "Access valuable information how to take care of your eyes." It also shows "Test Instructions" for Cataract Test and Blindness Test, and a "Start Test" button.
- Upload or take image:** The user is prompted to "upload photos or taking photos?" with options for "Camera" and "Galery". The screen also displays the same test instructions and navigation icons.
- Take Image:** The camera viewfinder is shown, with a large eye icon in the center and a camera shutter icon at the bottom. The screen also displays the same test instructions and navigation icons.
- Detecting:** A progress message "Wait a minute... Detection and classification of cataract types" is displayed over the camera viewfinder. The screen also displays the same test instructions and navigation icons.
- Result:** The final result screen shows a thumbnail of the eye image analyzed. The diagnosis result is "Immature". The screen also displays characteristics of cataract (e.g., blurred vision, sensitivity to light), treatment suggestions (e.g., medical consultation, use of assistive devices), and further advice (e.g., avoiding bright light). Navigation icons for Home, OptiLens, History, and Profile are at the bottom.

C242-PS469

bangkit

Blindness Select

Instruction

Test-1

Test-2...N

Result

 OptiLens
See the Future Clearly

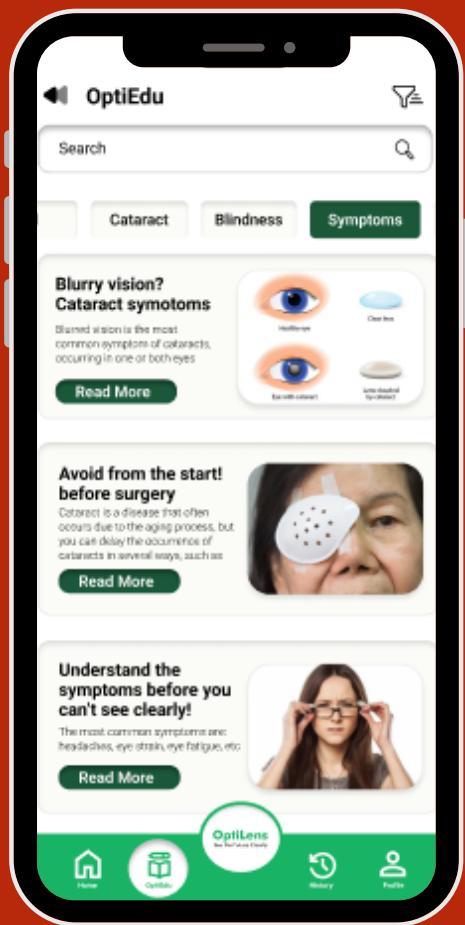
Mobile Apps Design

 NEXT

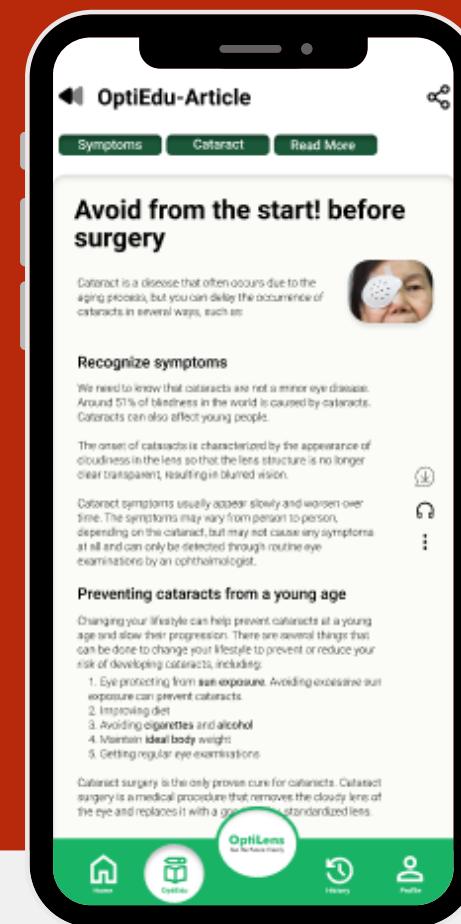
C242-PS469

bangkit

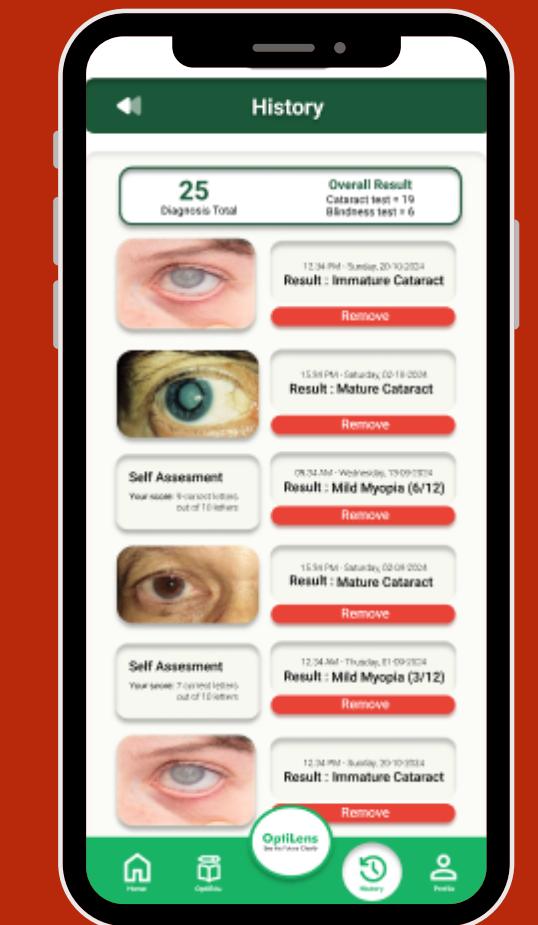
Mobile Apps Design



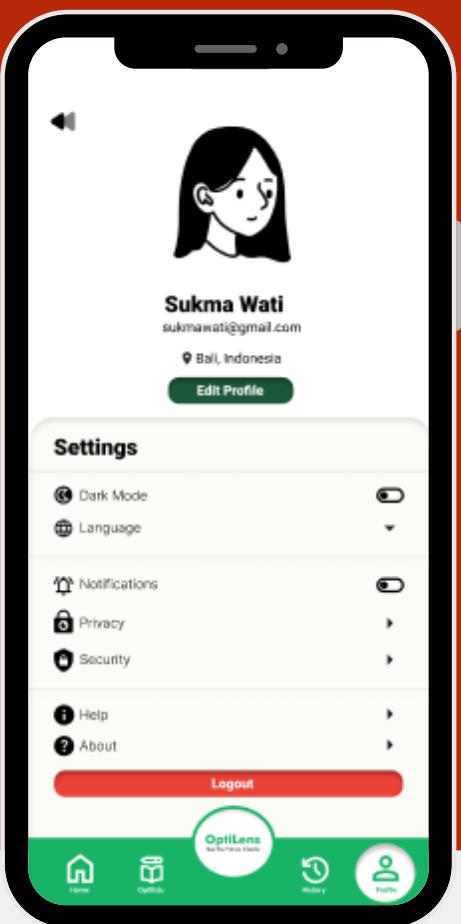
OptiEdu



Select Article



User History



User profile



Setting Profile

Person-Hours

Machine Learning - Team



Abd Rahman
Wahid
68 Hours



Salsabila
Putri
68 Hours



Muhammad
Hidayatul Fadillah
50 Hours

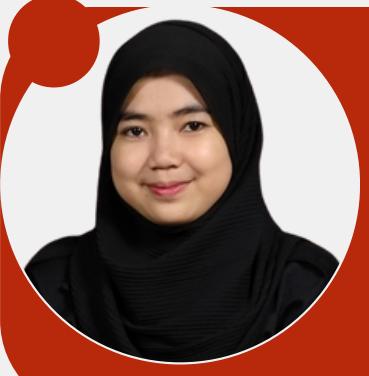
- **Optimal Data Preparation:** We structured the dataset into three categories (Immature, Mature, Normal) and applied data augmentation to enhance data variety and improve model robustness against overfitting.
- **Efficient Model Architecture:** Our CNN architecture, featuring convolutional, pooling, and dropout layers, effectively extracted features while minimizing overfitting risks.
- **Focused Training and Evaluation:** Using the Adam optimizer and categorical crossentropy loss, we achieved efficient convergence. Callbacks, like early stopping, ensured training stopped at optimal performance. Visualization of accuracy and loss trends facilitated overfitting detection.
- **Ease of Inference:** The model processes uploaded or webcam-captured images with high accuracy, making predictions seamlessly.
- **Cloud Integration:** Converting the trained model to .h5 format allowed smooth deployment on cloud servers.

Person-Hours

Cloud Computing - Team



Ahmad Faisal
60 Hours



Galbi
Nadifah
60 Hours

On the backend, we used the NestJS framework written in TypeScript to build a modular and extensible application architecture. The database used is PostgreSQL, which is reliable for high-integrity data management. To support deployment and scaling management, we utilize Google Cloud Platform and use Docker to package the application in an isolated and consistent environment.

Mobile Development



Ahmad Faisal
40 Hours

The app development process starts with designing the UI/UX prototype in Figma, translating it into functional components in Android Studio using Kotlin. The project is set up on GitHub for version control, using modern architectures like MVVM for scalability.

C242-PS469

bangkit

Thank You